EMERGENT PRODUCT DEVELOPMENT PROCESS STRUCTURES

by

PRISCILLA H. WANG

B.S. Mechanical Engineering
Massachusetts Institute of Technology, 1998

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Signature of Author.....
Department of Mechanical Engineering
May 5, 2000

Certified by.....
David Wallace
Esther and Harold E. Edgerton Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

EMERGENT PRODUCT DEVELOPMENT PROCESS STRUCTURES

by
PRISCILLA H. WANG

Submitted to the Department of Mechanical Engineering
On May 5, 2000 in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Mechanical Engineering

ABSTRACT

Advances in communications technology have introduced the potential to transform the product development process from a static system to one that evolves dynamically on a product-by-product basis. New design environments are providing a simulation and service marketplace that allows participants to flexibly define and change relationships, thus causing the development process to emerge with the definition and detailing of the product. The emergent nature of this new environment increases the need to dynamically monitor and potentially manage the resulting product development process structures. A software tool is designed to analyze the structure of product development systems. The tool extracts information related to process, organization, and design structure in the form of a Design Structure Matrix. As a part of the DOME (Distributed Object-based Modeling Environment) prototype system, the tool automatically analyzes the information flow structure in an integrated product design model. A case study concerning the design of an automotive door window system is used to demonstrate the tool and to explore the transformation from a static, centralized product development process to an emergent, distributed process.

Thesis Supervisor: David Wallace

Title: Esther and Harold E. Edgerton Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

The author would like to thank a number of people who have made this thesis possible. First and foremost my family for their encouragement and support. Professor David Wallace for his vision, guidance, and patience, and for giving me the opportunity to learn from this remarkable environment as an undergraduate and graduate student.

Thanks to all the members of the Computer-Aided Design Laboratory at MIT for providing a stimulating and entertaining workplace without which I would not have enjoyed my graduate years at MIT nearly as much. To Nick Borland for his patience and teaching in software development, Johnny Chang for his systems support, and Elaine Yang for her DOME support. To Shaun Abrahamson, Ben Linder, and Matt Wall for their DOME development. Special thanks to Jeffrey Lyons, whose assistance in the development of the plug-in, MGS project work, and companionship at Ford were invaluable. Thanks are also due to the rest of the Ford Five, Juan Deniz, Bill Liteplo, and Chris Tan, for their contributions to the MGS project and general camaraderie, and to Stephen Smyth for his assistance in many matters. And, of course, thanks to my faithful cadlab35.

Thanks as well to Peter Sferro, Bob Humphrey, Don Smith, Darrell Kleinke, and Al Clark for their help in understanding the MGS development process and in getting things done for the project.

For their support in getting through graduate school and especially the thesis times, I would like to say thank you to all of my Six Cents and to Wave. In particular, very special thank yous are due to Steve and Olympia (now you can really call me Master P!).

Work reported in this thesis was supported by Ford Motor Company and the National Science Foundation through the MIT Center for Innovation in Product Development, agreement Number EEC-9529140.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Problem Statement

Changes in information technology present the possibility of loosely formed, rapidly evolving product development teams and organizations. In contrast, most companies currently use a highly structured, static process model for developing products. However, increasingly complex products and distributed resources make it difficult to have the many heterogeneous and distributed resources involved managed in this centralized way.

In addition to part design, a multitude of tasks, such as thermal or FEA analyses, cost estimates, and marketing analyses, needs to be performed to develop a product. These tasks can be thought of as *services* provided by the various development resources. The developing of products becomes a process of combining pooled resources and seeking other resources that are needed. This environment, with resources exchanging design services to form a product development system, can be viewed as a *service marketplace*. The Distributed Object-based Modeling Environment (DOME) concept being developed in the MIT CADlab creates such a service marketplace environment for modeling the characteristics of new products.

The transformation of the traditional product development environment into a service marketplace raises many new design process issues. The need to coordinate the service buyers and sellers may cause the structure of the product development process to emerge as the product is being designed. Product design choices may directly affect or even drive the process structure, or vice versa. The transformation of product development into a marketplace implicates a more distributed design process without an overriding structure.

Highly structured traditional product development exhibits many properties of a poorly functioning market. Centralized control in development often results in poorly advised decisions and inefficiencies in the use of resources. In contrast, an emergent system

environment for product development could prove extremely beneficial. Emergent systems allow individual parties to complete their tasks efficiently and assist in the effective allocation of resources.

An information technology-based service marketplace offers the potential for monitoring and analyzing the structure of the service exchange network, or process, as it evolves. A system analysis tool was developed to assess the design and development process structure of product models in DOME (Distributed Object-based Modeling Environment). The tool was used to study design models in a service marketplace setting and to conduct comparative tests. A case study with an automobile manufacturer to design Moveable Glass Subsystems provided an industrial example for testing of the tool. The modeling of the MGS in DOME and interaction with both the OEM and its suppliers allowed in-depth study of the MGS product design structure, the existing development process structure, and new possibilities presented by the service marketplace environment.

1.2 Overview of the thesis document

Chapter two gives context for the work presented in this thesis. It begins with a brief overview of typical sequential product development. The chapter then discusses the need to understand the product development process and reviews some methods of process analysis. Then practices and criteria for selecting with suppliers are discussed.

Chapter three explores the strengths and weaknesses of emergent systems. Observations are made of emergent systems and their importance, characteristics, and properties. From these observations a list of desirable attributes for an emergent system environment is generated. Finally, the notion of product development as an emergent system is considered.

Chapter four describes the Distributed Object-based Modeling Environment research software and the new context for product development that it gives rise to, a simulated product development service marketplace. Relevant features of the software functionality are explained as well.

Chapter five details the software tool designed to analyze product development process structures built in DOME. Use of the tool is described for model navigation, understanding system element dependencies, and product development organization design.

Chapter six describes a case study of a vehicle door window subsystem used to test the research software and tool. The chapter first explains the development process as it existed at the OEM and then as it was represented in the research software. The process analysis tool is also tested and compared to interview-generated Design Structure Matrices. The DOME service marketplace environment is then evaluated, and finally the transition of changing from a centralized product development process to a more evenly distributed, emergent process is discussed.

Finally, Chapter seven reviews the material presented in the thesis, draws some conclusions on emergence in product development, and proposes further work.

2 BACKGROUND

2.1 Traditional Product Development Process

The product development process can be defined as "the sequence of steps or activities which an enterprise employs to conceive, design, and commercialize a product" (Ulrich and Eppinger 2000). The general characteristics of successful product development include good product quality, product cost, development time, development cost, development capability (Ulrich and Eppinger 2000).

Because product development is complex, lengthy, and involves many people, it is typically desired to have a well-defined product development process. A well-defined process is generally comprised of milestones, phases, and clearly articulated development tasks that together act as a master plan for the project (Ulrich and Eppinger 2000). To realistically accomplish this, often one entity must oversee the entire process.

In practice today, product development within organizations is commonly forced to follow a sequential, rigid process. At times processes that must be followed are obsolete, or tedious phases and checklists must be completed in order to satisfy company policies and directives. Progressing to the next product development stage may take extra time and effort, and the act of progressing to the next stage may not really address the critical issues of the product design. Also, development activities may be completed at different rates so that some activities may have to wait unnecessarily long to be begun in the next development stage.

Communication barriers between product development participants introduce significant time lags into the development process as well. In a sequential design process, the effects of communication time on the overall development schedule become magnified. Consequently, in a rigid, sequential development environment it is often very costly to perform design iterations and explore many options.

Huang and Mak (Huang and Mak 1997) compile an extensive list of other problems associated with sequential development processes. Table 2-1 lists most of the pertinent issues addressed in this thesis.

Table 2-1: Typical problems with sequential engineering, adapted from Huang and Mak (1997)

- Sequential activities result in protracted cycle times
- Intermediate milestones receive a disproportionate amount of focus
- Too many checkpoints exist, and there is too much waiting to be checked
- Communication is inadequate, inefficient and/or ineffective
- Information becomes lost or obsolete
- Scarce resources are wasted in fire-fighting, progress chasing, and making changes
- 'Inertia' is too high for product development system to be responsive
- Unnecessary technical complexity exists in products, processes, and systems
- Single-feature optimization leads to sub-optimal solutions
- Poorly structured product development leads to poor coordination
- Problems are discovered too late, resulting in panic and leading to 'quick fix' solutions and compromises and long and costly rework loops
- Islands of expertise exist, and human skills are narrow
- Management processes are isolated
- Hierarchical structures lead to a situation where managers think and make decisions, and contributors work and enact decisions

As an improvement over sequential development, concurrent product development has become the goal of many development processes. Concurrent product development involves the completion of product development tasks in parallel. Thus, it is intended to lessen time lag effects, provide better feedback, and eventually improve quality. Concurrent product development is practiced to some degree in industry today, but its benefits have not yet been fully realized (Yan and Jiang 1999).

Despite the movement towards concurrent product development, it is still believed that the concurrent process should be well defined, well proven, and relatively static. The idea of a changing process, especially one that changes during a product's development, is generally not accepted.

2.2 Need to Understand the Process

Over the past few years, the importance of understanding and improving the efficiency of the product development process has become increasingly recognized. Problems and risk in the product development process have been found to significantly contribute to the failure of new products, and management and control of the development process are key to minimizing them (Ahmadi and Wang 1999). In order to minimize problems and risk, the product development process must be managed. Ahmadi and Wang state that, "Like production processes, design processes also need to be monitored and controlled in order to achieve high design efficiency and quality," (ibid). The importance of a good product development process to reducing product development cycle time has also been affirmed (Griffin 1997).

Other sources agree. In product development today, especially with regards to concurrent engineering, there is a recognized need to "capture, manage, coordinate and utilize the CE environment's constantly evolving data, knowledge, and processes," (Prasad, Morenc, and Rangan 1992). However, a balance of control needs to be maintained. Ahmadi and Wang note that, "Both over-managed and under-managed development processes result in lengthy design lead time and high development cost" (Ahmadi and Wang 1999). Accordingly, the effort devoted to overseeing the development process and the degree to which resources are allocated at each stage must be moderated in order to maximize efficiency.

The possession and availability of system knowledge is key to controlling the process. In a distributed product development case study focused on disseminating information, the importance of informing participants of the workflow and process was highlighted (Hameri and Nihtila, 1997). It was found that making the information widely visible

encouraged and facilitated self-organization among the participants and further improved the process as well. Crabtree, Fox, and Baid likewise maintain that solving the problems of collaborative design must begin with gathering information about the system (Crabtree, Fox, and Baid 1997). System interactions and dependencies among design tasks must be understood in order to improve the process (Wang and Jin 1999).

2.3 Product Development Process Analysis Methods

A number of efforts have been made to analyze product development processes, as are highlighted in this section.

2.3.1 The Design Structure Matrix

The Design or Dependency Structure Matrix (DSM), attributed to various sources including James Rogers (Rogers 1997) and Steven Eppinger (Sabbaghian, Eppinger, and Murman 1998), provides a system-level visualization of the relationships between product development elements. Here we will discuss the more recently developed DSM by Steven Eppinger. The DSM maps dependencies between elements in a bi-directional graphical format. Four different types of elements have been most commonly analyzed with DSMs including design parameters, product components, development activities, and individuals or teams.

Construction of DSMs typically requires a lengthy process. It involves interviewing product development participants, determining a list of tasks or parameters to be represented, asking about the element relationships and strengths of interactions, entering the information into a matrix, and verifying with the interviewees for content accuracy (The MIT Design Structure Matrix—DSM Home Page 2000). A Microsoft Excel macro is commonly used for the information entering and analysis. The human interaction involved in these steps requires a significant effort overhead that can be problematic, particularly for larger projects (Sabbaghian, Eppinger, and Murman 1998). Moreover, the human interaction can also result in somewhat subjective and potentially faulty DSMs. "The *success* of the DSM method is determined by an appropriate system decomposition and by the accuracy of the dependence relationships collected. Therefore,

it is *vital* to carefully decompose the system under study into meaningful system elements," (The MIT Design Structure Matrix—DSM Home Page 2000).

Analysis of the examined the DSM element dependency mappings and feedback patterns are used to suggest ways of making product development processes more efficient. Resequencing and reorganization of tasks or other elements in the development process can result in reduced design cycle time, reduced iterations, and cost and product performance improvement (Dong 1999, Smith and Eppinger 1998). However, process data gathering, DSM generation, and restructuring analysis generally are completed after the product has been developed or in a mature product development environment. Thus, the analysis usually can only benefit subsequent projects, not the project that has been analyzed. Furthermore, if the product development process changes during the creation or use of the DSM, the DSM risks becoming obsolete because it can only present a static view of the process (Whitney, Dong, Judson, and Mascoli 1999).

2.3.2 Automatic process analysis methods

Other methods have been used to generate product development system analyses while avoiding the problems of gathering system information via interviews. These methods automatically extract process information from computer-based product development applications.

Groupware is a means of providing group access to shared file repositories, discussion forums, and communication facilities. The activity surrounding Groupware in product development projects has been used to analyze the development process (Orlikowski 1995). However, examining the use of repositories, forums, and communication facilities renders only a narrow view of product development activities.

The product development software VisionManager (Fruchter, Reiner, Leifer, and Toye 1998) interprets a shared product model, gathers information, analyzes and evaluates product development behaviors, explain results to product development team. It then routes notifications for proposed changes by augmenting shared graphic design models with team members' ontology, design intents, and responsibilities. VisionManager also attempts to analyze the product development process by capturing dependencies at different levels of granularity and for different versions over time. However, VisionManager's functionality is limited to a database management system with CAD and e-mail capabilities, and thus prevents large portions of product development from being analyzed.

2.4 Supplier Selection

Developing products often requires procuring components or other design or analysis services from other companies. To procure the best parts or services, the OEM has to choose its partnering companies carefully. The criteria that OEMs would like to use to select suppliers generally includes consistency of quality and delivery, reliability, relationship with the OEM, flexibility, price, and service (Choi and Hartley 1995). However, supplier selection often involves a lengthy, rigid process with substantial of negotiation. Some OEMs may employ bidding processes while others follow specific, detailed selection processes. Due to these lengthy selection processes, supplier choices sometimes are made not based on the desired selection criteria. Time constraints in the development process and lack of timely and complete analysis capabilities may force supplier decisions to be made before there is a thorough enough understanding of the alternatives. For example, supplier choices may be made based on long-term relationships rather than actual development capabilities (Choi and Hartley 1995). Consequently, a serious need has been recognized to better correlate appropriate performance measurements to supply chain decision variables and have a more integrated approach to system design (Beamon 1998). Thus OEMs may more intelligently analyze their options and make the best decisions.

3 EMERGENT SYSTEMS

In this chapter a general understanding of emergent systems is developed. As the emergent system of the free marketplace has been studied in depth, the field of economics is used to generate a list of emergent system principles and characteristics. These characteristics are then related to product development systems.

3.1 A Proposed Definition

In this section a definition of an emergent system is synthesized from various sources.

A straight definition of the term *emergent system* is taken from a dictionary. Dictionary.com describes a system as a group of interacting, interrelated, or interdependent elements forming a complex whole; a network of structures; or an organized set of interrelated ideas or principles (Dictionary.com 2000). The same source defines emergent as coming into view, existence, or notice; or rising above a surrounding medium (Dictionary.com 2000). Thus, an emergent system is a system composed of dynamic elements with a dynamic structure. In general, the elements comprising the system compete with one another in some manner.

Emergent systems are constantly changing and evolving. They are in "the state of being in continual process, never arriving but always in transition," (Truex, Baskerville and Klein 1999). Emergent systems may exhibit temporal regularities, but they are never static (Truex and Klein 1991). The conditions of and relationships between the system elements continually change. Moreover, the evolution of emergent systems follows no predefined path (Truex, Baskerville and Klein 1999). Because emergent systems do not have explicit central control, their development cannot be fully directed.

Many types of emergent systems exist including biological systems, ecological systems, and any organic systems or systems driven by human behavior. Social and political systems, economic systems, linguistics, and companies all are examples of emergent systems. When systems grow extremely large, it is no longer feasible or practical for

them to be managed strictly by an overriding power or to adhere to any rigid structure. Thus, it may be more natural for systems to emerge.

3.1.1 System structure

The structure of a system may relate to many of the system's attributes. An online dictionary defines structure as "the way in which parts are arranged or put together to form a whole," or, "the interrelation or arrangement of parts in a complex entity," (Dictionary.com 2000). In an emergent system, structure then refers to "the static aspects of a system that are usually observed by its apparent regularities," (Truex and Klein 1991). Thus when a snapshot of an emergent system is taken in time, the system structure describes the temporal seeming relationships and dependencies between the elements.

3.2 The Importance of Emergence

Emergence in systems is important to allow the exploration of different avenues. In order for systems to evolve to better states, various combinations and options must be tried. Improvements and progress in systems typically arise out of such experimentation and unplanned events (Coleman 1999). Also, changes in systems in turn create new avenues and opportunities for further development (Inayatullah 1994).

Evolution and adaptation of a system is crucial to its survival. As explained by Inayatullah, static system structures and rigid relationships between the system elements can lead to the demise of systems (Inayatullah 1994). Emergent complex systems prevail because they "harbour behaviour which is the most flexible and adaptable." Because emergence allows a system to best respond to changes in the environment, it is a critical property during periods of volatility.

3.3 Characteristics and Properties of Emergent Systems

As previously explained, emergent systems by definition are dynamic. Due to internal and external forces, the elements making up the system are always changing, as are the

relationships between the elements. Competing elements of the system maintain the system dynamism. The competition causes the elements to search for ways to improve the status quo and forces new and increasingly better options to be available. As the best options are naturally selected, the competing elements are forced to adapt and to find new solutions (Inayatullah 1994). Thus, the system and structure emerge and evolve.

Adversarial forces inside and outside of a system introduce conflict into the environment. In order for a system to adapt to its changing environment effectively it must be in a constant state of flux. To settle into a steady state would be detrimental to the system (Inayatullah 1994). Inayatullah asserts that a robust emergent system must have a structure that is so fluid and elements that change so much that the system is almost to the point of chaos (Inayatullah 1994).

However, a completely chaotic environment is not conducive to system progress. The best conditions of all worlds combine chaos and order in a self-regulating and self-learning system (Inayatullah 1994). Inayatullah explains that, "Evolution thrives at the at the edge of chaos, where neither chaos nor order is dominant. This balance allows for gradual controlled change, where flexibility can emerge." (Inayatullah p. 694)

The economic world serves as a good example of an emergent complex system. Composed of the many competing elements of individual people and companies it is a highly complex, continually changing system. The field of microeconomics involves the study of the buyer and seller elements and how they interact within the marketplace. We can draw from the knowledge and understanding of economics to examine the properties of an emergent system.

In the marketplace, buyers and sellers must act *independently* in order for markets to function efficiently (Pindyck and Rubinfeld 1998). The individual elements must conduct their own transactions in their own interests without much interference from a central power. Although a centrally controlling power, like the government in the economic world, could dictate the system transactions and allocate the resources, it would

require that the central power maintain extensive knowledge of the system. Independently acting parties requires the least amount of information.

According to the First Theorem of Welfare Economics, independent actions within a competitive marketplace will result in the occurrence of the right transactions and an efficient allocation of resources (Pindyck and Rubinfeld 1998). Moreover, as parties act in their own interests they bargain to their mutual advantage (Pindyck and Rubinfeld 1998).

In order for the system to continue to emerge and evolve, the system must remain competitive. Minimal intervention may be required in order to preserve competition.

Economists have identified two other cases in which outside intervention can be beneficial to the total welfare of participants in a competitive market (Pindyck and Rubinfeld 1998). The first case is when the costs or benefits of consumers' or producers' actions do not become evident in the market. The second case is market failure, when the price of the service does not provide enough information for consumers to make decisions that maximize their utilities. Accordingly, intervention becomes advantageous when the consequences of decisions or actions are not understood. Access to complete and accurate information about the market is critical to efficient system operation.

Finally, at times resources must be reallocated between buyers and sellers in a market. Although often the effort involved in repossessing and redistributing the resources causes inefficiency, economics tells us that the actual new allocation of resources itself does not have to conflict with market efficiency (Pindyck and Rubinfeld 1998). Hence, if reallocating resources takes little effort then it can improve the system's efficiency. In this case as well intervention would be beneficial for the system.

From these examples we can develop a summary of the attributes of an emergent system, as listed in Table 3-1. Also included are attributes that make a system efficient.

Table 3-1: Characteristics of an emergent system environment

- Majority of the (trans)actions are performed by the individual elements
- All elements are given a fair chance to compete for business
- Control and knowledge ownership are distributed among the elements
- Transaction costs must remain low
- System structure and processes are not predetermined and different options are allowed to be explored at little or no cost
- Sufficient understanding of options and their consequences are available to the individual elements
- Accurate information about the market, its structure, etc. are available to all
- Minimal regulation of the system is imposed as needed to keep competition fair, information disseminated, and resources efficiently distributed

These attributes are necessary in an environment for an efficient emergent system to subsist. In short, the best emergent system involves minimal checks and balances. There must be enough freedom to explore different options, create new solutions, and allow changing structures to develop. The system must be allowed to evolve in response to surrounding forces so that it can naturally evolve to adapt well. At the same time, efficiency and some sense of structure need to be maintained so that the individual elements can understand the system, navigate through it, and operate within it intelligently.

3.4 Product Development as an Emergent System

On many different levels product development acts as or has the potential to become an emergent system. To begin with, the physical product itself is emergent. The parts and parameters of a product form a system of interdependent elements that change throughout the product's development. The various elements affect and compete with each other both directly and indirectly through the system. Competition between these elements manifests in quantifiable measures such as space, manufacturing cost, weight, and thermal output, as well as in other performance attributes that are important to product developers.

The product development organization, comprising the different parties involved of individual people, groups of people, supplier companies, and the OEM, together have the potential to comprise an emergent system. At different stages during the project, different parties are utilized to complete the various development tasks. What parties are active at a given time may change often according to the project stage, the available resources, and unforeseen problems that arise. Additionally, competition between the different supplier elements for business from the OEM certainly exists. Similarly, different groups or individuals within the same company in a less obvious sense compete for task responsibility and resources. It is important recognize that collaborative design engineering is a social process (Crabtree, Fox, and Baid 1997), and consequently the process tends toward emergence.

In a more abstract sense, the process as a whole involving what product development tasks are accomplished, the order the tasks are completed, and how the tasks affect one another form another emergent system. The individual and groups of tasks compete for the limited time, resources, and attention of the product development participants.

Product development thus combines many competing elements: parts with parameters being compromised to satisfy design specifications and performance measures; suppliers vying to win business and negotiate good prices; and development tasks competing for monetary resources, project schedule time, and attention. It is difficult to distinguish between these different types of system elements, and even more so to separate them from one another as they overlap realms and are interdependent. In fact, products are best developed when the types of elements do not have to be separated but can be analyzed and (optimized) together. Solutions to the system must involve the integration of people, procedures and technologies (Crabtree, Fox, and Baid 1997).

In addition, product development is highly subject to the influences of other emergent systems that are involved in a product's development such as economic systems, social systems, and organizational networks. Thus, product development in many aspects shows the characteristics an emergent system. However, current rigid development processes tend to inhibit the emergent properties of the organization and thus its ability to respond to change.

As described in sections 3.2 and 3.3, product development could benefit from following more of an emergent system model. Allowing the individual development participants to complete their design tasks and the system structure to be fluid enables the best design choices to be made and the resources to be most efficiently allocated. An emergent product development system could truly realize concurrent product development with and parties interacting directly with each other and tasks being completed as needed independently of others.

In light of business strategies, the threat of competitors, and changing technologies, it is imperative that a product development system be able to adapt to its changing surroundings. Organizations must be nimble and responsive to changing market and customer needs in order to compete. Hence the fluid structure of an emergent system model could be critical to a product development organization's survival.

3.4.1 Product development system structure

Product development system structure in this thesis will be defined as the relationships between the various product development system elements. It involves the product development process intertwined with the product development organization. Structure has multiple dimensions in the different types of elements involved—individuals, groups of people, product parts, design parameters, performance measures—and in the interactions between them—the information exchanges, the order of tasks, and how decisions affect other decisions directly and indirectly.

In the context of an emergent environment, product development system structure must continually change. It is this freely developing structure that allows the system to adapt and survive. However, the changing structure and resulting process must be monitored and managed to promote design process efficiency.

4 THE DOME SERVICE MARKETPLACE ENVIRONMENT

The Distributed Object-Based Modeling Environment (DOME) provides the opportunity for product development systems to become emergent and to realize the potential emergent system benefits.

4.1 Functional Description of DOME

The Distributed Object-based Modeling Environment (DOME) research software creates a World Wide Web-based environment for bringing together various product development participants and activities. By providing the necessary software to connect individual computer-based models and simulations, DOME allows users to create large, integrated product models.

DOME allows users to exchange information between heterogeneous computer applications that otherwise would require human interaction and communication. DOME supports the integration of CAD software (I-deas, ProEngineer, SolidWorks), spreadsheets (Microsoft Excel), databases (SQL), finite element analysis (MARC-Mentat, ANSYS), environmental life cycle analysis (Ecobilan), neural network mathematical estimations (Matlab), life cycle analysis (TEAM), and any custom code compiled as dlls. Additionally, the distributed capabilities of DOME allow users to work with models physically located on different machines, on different local networks, and around the world. DOME manages the information transfer necessary to propagate design changes to all of the product development parties, keep the models updated, and show system-wide effects. Thus, product development participants can engage in their usual work tasks using their appropriate tools but more rapidly and efficiently pass information and receive feedback from other participants and tools.

Individual DOME users can choose to provide or subscribe to the services of other participants or combine and interrelate design parameters, development tasks, people, and computer applications together.

In addition, the resulting DOME integrated product model can be subjected to various analyses. Various DOME plug-ins are available to evaluate and score overall designs, perform searches or optimizations for the best design configurations, and further support trade-off analysis and decision making.

Use of the DOME environment results in a rapid communication network connecting the many distributed participants in a product development process. This rapid network has the potential to drastically decrease the time to complete a single design iteration as well as the time to bring a product to market (Wang 1998).

The simple steps to connect individual models or simulations together through DOME involve defining the model's interface to the DOME integrated product model system. The user needs only to specify how the individual model should interact with the rest of the system in terms of what pieces of information or services it requires from and can provide to others. This simple process of *publishing* is meant to follow the natural work flow of the product development participants. (Senin, Wallace, and Borland 2000)

4.2 Explanation of DOME-related Terminology and Concepts

Most of the following DOME terms are explained fully in "Distributed Object-based Modeling in Design Simulation Marketplace" (Senin, Wallace, and Borland 2000). Simpler definitions, adequate for understanding the discussions in this thesis, are included here.

In the DOME marketplace environment, a **service** refers to the "result of an action performed by an individual, computational tool, or organization" (Wallace, Abrahamson, Borland 1999). It may be defined as the provision of information by one party to another in the form of an analysis, making a design choice, or otherwise passing on information. The concept of a DOME service corresponds to services as they exist in the product development world. A DOME service involves the creation and interconnection of DOME service-objects.

A **service-object**, or DOME module, refers to the basic building block in a DOME model. All other objects in DOME are special types of service-object or are built of service-objects.

Third party models refer to computer simulations or other types of models built in non-DOME software applications. Third party models are used independently of DOME as a part of the normal workflow of product development participants. Use of third party models constitutes the performing of a service.

One or more service-objects and third party models connected in the DOME environment together form an integrated **DOME model**. DOME models may also include additional decision support or other analytical objects. A DOME model can behave as a single large model responding to changes as one entity.

In order to create DOME models with distributed services and third party models, the services and models must be **published**. Publishing refers to the process of defining an interface (what informational inputs are needed and what informational outputs can be provided) to a service and making the service available to other DOME users on the internet.

Connecting to and using a published service is referred to as **subscribing** to the service. A DOME model can subscribe to multiple services, and a service can be subscribed to by multiple DOME models at once.

Within a DOME model, service-objects and third party models are connected via **relation** service objects. Relations define mathematical dependencies, or how one or more service-objects affect another. Relations also specify dominance, or which service-objects drive other service-objects.

An **alias** service-object behaves as a copy of another service-object. A change in either the original service-object or its alias will cause a change in the other. Thus, an alias relationship is bi-directional with no dominance between the service-objects.

A **container** service-object is an object that can contain other service-objects. This encapsulation capability in DOME helps manage the organization of and relation building between other service objects. DOME models thus can be viewed at various levels of encapsulation, as containers can be viewed by themselves closed or open with their contents displayed. Containers can also hold other containers.

A **DOME server** refers to a machine that runs all of the DOME server software and can hold one or more DOME model files. Only one DOME server can be run on a given machine. DOME servers control the logging in permissions of and can host multiple DOME clients.

A **DOME client** is used to log into a DOME server and run a DOME model. A DOME client is run through a standard WWW browser. Multiple DOME clients logging into the same or different DOME servers can be run on the same machine.

4.3 Service Marketplace Environment

DOME is readily understood as a service marketplace for product development when compared to the economic world. Many economic concepts have parallels in the DOME environment. The performing of transactions and exchange of good or services in the economy is analogous to subscribing to services in the DOME world. Competing for business in the market parallels publishing services and making them available to other DOME users. Buyers and sellers and economic resources match product development resources such as people, time, and money.

In the increasingly complex and distributed process of developing new products, DOME provides an environment in which the distributed and heterogeneous participants,

components, parameters, and tasks of product development can freely interact. Independent of physical location or software media, the participants can exchange information on all levels, receive constant updates, and maintain knowledge of the project state. A central holder of the system knowledge and a single absolute view of the system is not needed. The distributed architecture of DOME allows the system elements to "interoperate at a global level while maintaining control at a local level...with the management qualities of a centralized system, but the responsiveness of a locally autonomous system," (Abrahamson, Wallace, Senin, Sferro 1999). Thus, the many individual elements involved in the product development process are allowed to participate in the marketplace.

In the DOME environment, the whole structure of the product development process does not need to be determined prior to modeling a product in or using the capabilities of DOME. Moreover, there is no required or no prescribed structure for an integrated DOME product model.

The various product development participants can be exchanged into and out of the DOME product model as desired. Participants vying for a particular role become readily interchangeable. Similarly, whole groups of services can become interchangeable with other groups of services. Mimicking how one supplier and its provided capabilities might be swapped for another, DOME supports the engaging and disengaging of parts of the product development process on any level through catalog service-objects (Senin, Wallace, and Borland 2000). Catalogs allow single service-objects, a group of service-objects, or a group of groups of service-objects to be swapped.

Relation-objects allow connections to be formed easily and extensibly between service-objects in DOME models so that the entire DOME model is "plug-and-play." Because of the ability for individual participants to easily switch in and out and connect different services, the modeling and resultant product development environment can become dynamic, changing, and emergent. Furthermore, by making publishing, subscribing to, and swapping services and parties easy, the transaction costs are kept low thus supporting

a competitive environment. Easy comparison of design options helps maintain competition for healthy, changing emergent environment. Furthermore, as the time required to communicate information decreases, the time required to negotiate design decisions, complete design iterations, and make changes also decreases. This reduction in communication time allows more project schedule time and frees up human resources for greater exploration of design alternatives.

Moreover, because DOME product models can evolve into any formation and can functional at any stage of the system's development, DOME supports the unmediated, free-form building of models. It thus supports the emergence of model structure from its earliest stages of growth through major changes into a larger, more developed working system.

4.4 Simultaneous Product and Process Development

Depending on what portions of the product development system are modeled in DOME, designing the product may inevitably affect the product architecture, product development organization, and process. If DOME is used to exchange information and design the product, and if the relevant tools and people are incorporated into the modeling environment, then the DOME system *is* the product development process (Wallace, Abrahamson, and Borland 1999). As design parameters are changed and different product development participants and services are brought in and taken out of the product development system, the state of the DOME integrated product model represents the design of the product. The exploration of different design alternatives and reallocation of design responsibilities to design the product shapes the process at the same time. Thus, designing the product and the process no longer are separate issues.

4.5 Need to Understand and Direct the Emergent Structure

In an emergent product development environment there needs to be some way for the individual parties to understand and manage the process. The product development

participants may be familiar with the local scopes of their normal operation, but as discussed in section 3.3 they also need to understand rest of the system in order to act effectively. Particularly in an emergent environment, the system structure may change often in terms of task ownership, where information comes from, or what services are being subscribed to. Organizations, suppliers, and other groups may also change. Participants thus must have some way of navigating through the emerging system.

Additionally, participants need to be aware of available and requested services. Service providers and subscribers must be able to locate each other in order for transactions to take place. Furthermore, participants must understand their different options as well as the implications of their actions in order for serious competition to exist.

In such an emergent environment some measure of control on the growth and structure of the system must be retained. To preserve competition, redistribute resources, or reorganize processes, the mediating powers must understand the system structure. Management of the product development process and resources "involves obtaining an ongoing, detailed, timely understanding of the complex transactions," (Wallace, Abrahamson, Borland 1999).

5 DOME MODEL STRUCTURE ANALYSIS TOOL

The DOME Model Structure Analysis Tool provides a way to understand product development systems when modeled in DOME. The MSA aids navigation through the integrated product models and makes available centralized product development system knowledge for a distributed system.

The Design Structure Matrix (DSM) visualization is used to analyze the product models as an easy, compact way of understanding the system element interactions. This MSA tool, particularly with regard to the visual design and user interface, builds on previous work done by Shaun Abrahamson (Abrahamson 1999).

5.1 Plug-in Use

5.1.1 Building a Model Structure Analysis

To use the Model Structure Analysis plug-in the user must add a Model Structure object into a DOME model. A Model Structure object can be added into any container of a DOME model, regardless of the scope of the analysis to be performed. As shown in Figure 5-1, the user follows "Add" from main menu, then "Analysis" from the next menu and selects "Model Structure."

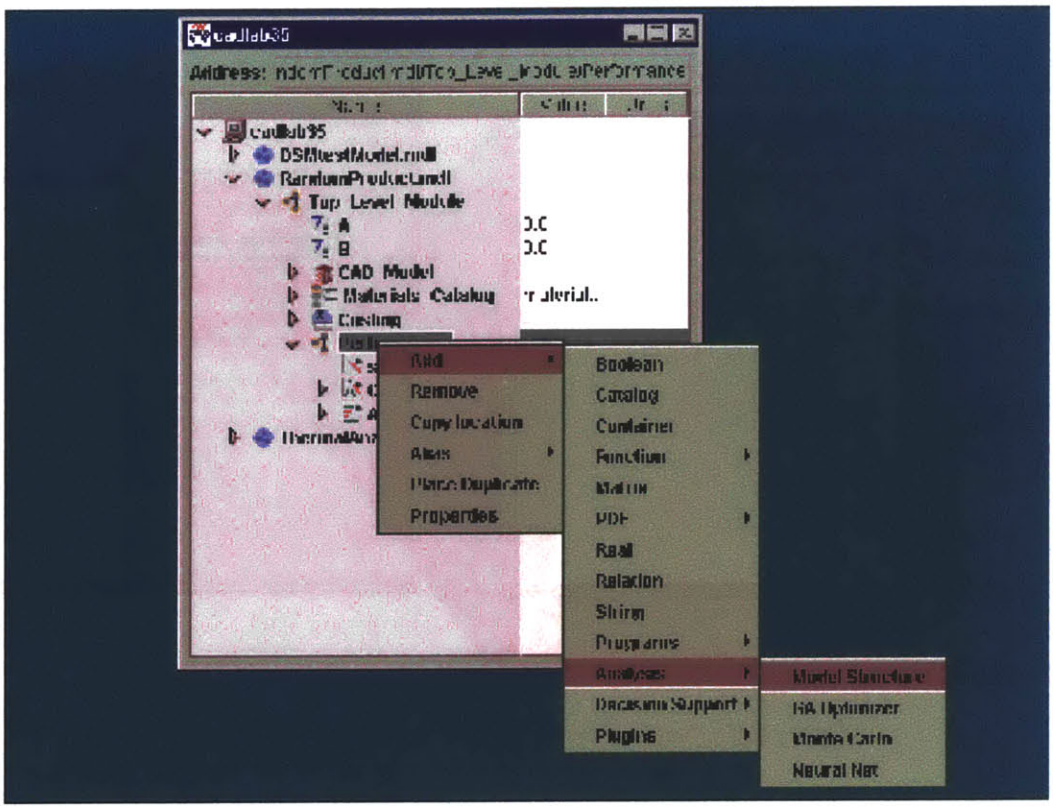


Figure 5-1: Adding a Model Structure Analysis Object into a DOME model

A new object with the default name "ModelStructure" will appear in the DOME model. The user then must open the graphical user interface for the Model Structure object by double-clicking on it. The container inside which the model structure analysis will be performed must be selected and its location copied as shown in Figure 5-2.

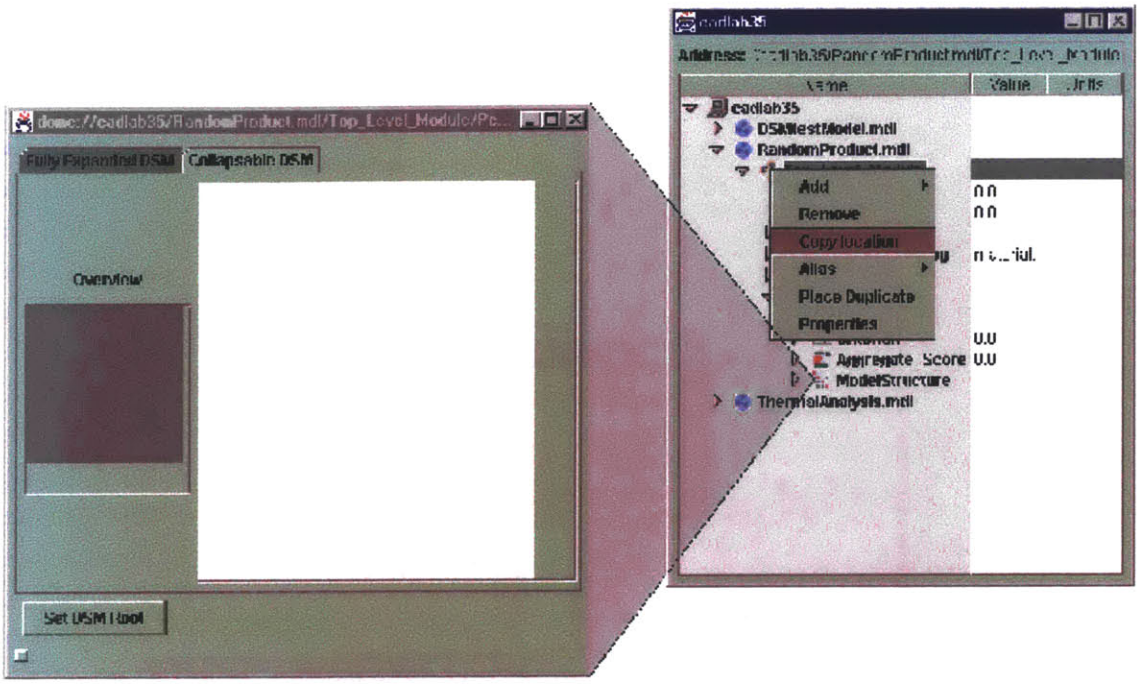


Figure 5-2: Setting up a Model Structure Analysis

A model structure analysis can be performed on any portion of a DOME model within a container module. The location of the MSA object has no bearing on the container to be copied. Once the desired container location is copied, the "Set DSM Root" Button must be pressed. The plug-in will then search through the DOME model to find dependencies and build the model structure analysis as shown in Figure 5-3.

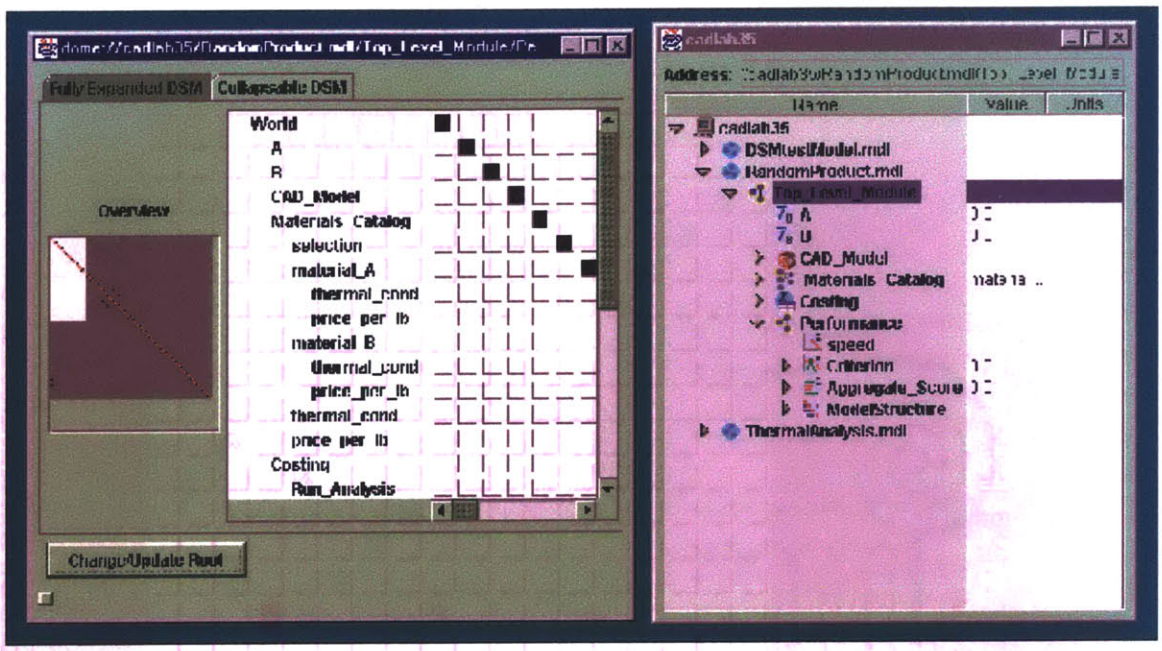


Figure 5-3: Completed Model Structure Analysis (MSA)

Two views of the model structure are provided. The view on the "Fully Expanded DSM" tab shows the model portion in the analyzed container fully expanded with the corresponding DSM. The "Overview" window to the left of the DSM contains a compacted view of the DSM and tracks with a white rectangle to where within the DSM the view is currently scrolled. The view on the "Collapsible DSM" tab shows the model in a tree view with a corresponding DSM that expands and collapses expands and collapses so that more specific portions of the model can be examined at a time. The tree view DSM is shown partially expanded in Figure 5-4.

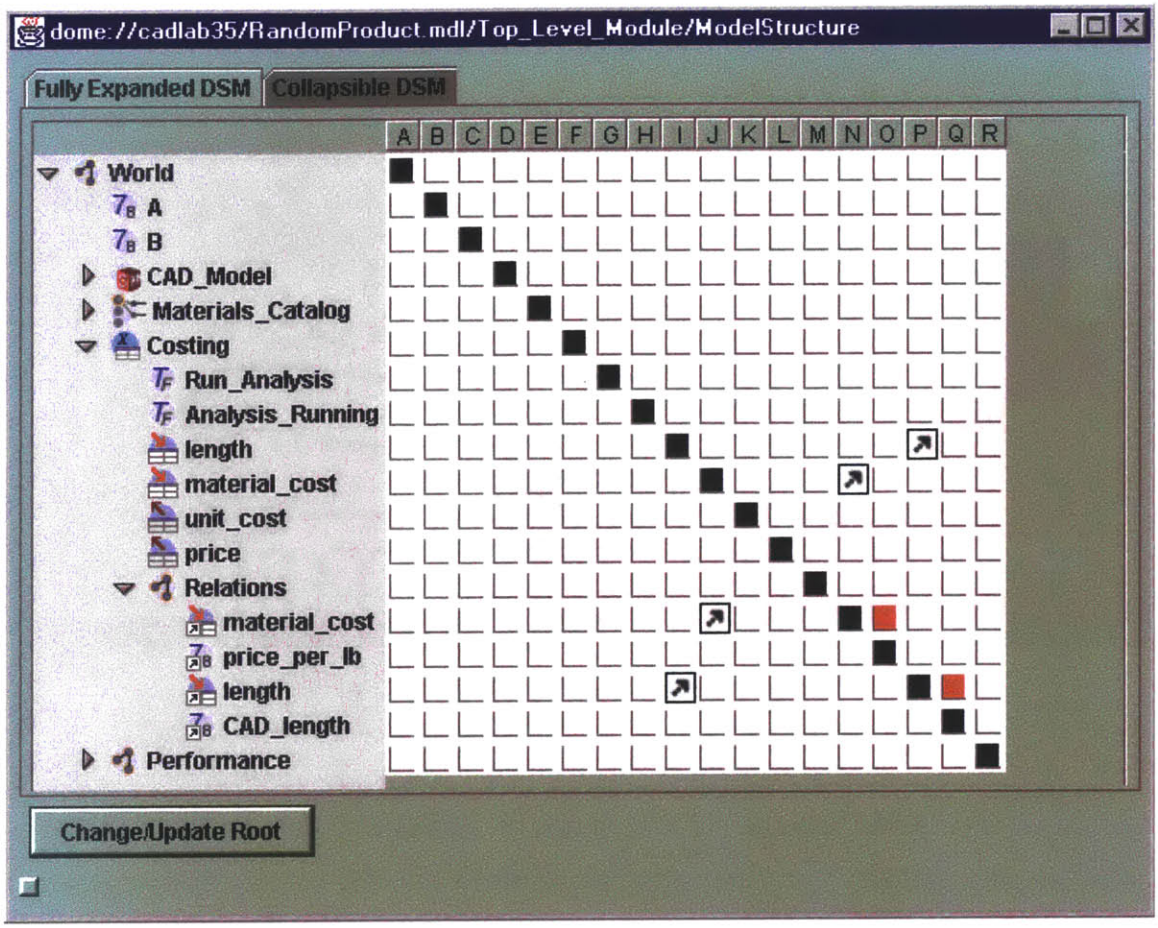


Figure 5-4: Tree view of a Model Structure Analysis

The same MSA module can be used to analyze within different containers by copying the new container's location and pressing the "Change/Update Root" button. Similarly, if a change occurs in the DOME model, the MSA can be updated by copying the container again and pressing same the button.

5.1.2 Reading a Model Structure Analysis

In a MSA DSM, black squares block out a module's dependency on itself and mark the diagonal of the matrix. Red squares signify a unidirectional relation dependency. Squares with an arrow icon signify bi-directional alias dependencies. Yellow squares signify aliases that are not mapped to any object or that are mapped to service-objects outside of the MSA scope.

To understand what other objects a given object is dependent on, the user reads across the object row. In Figure 5-5 the example DOME product model RandomProduct.mdl contains a red square denoting that the Excel cost model module "material_cost" is driven by the catalog container selection module "price_per_lb."

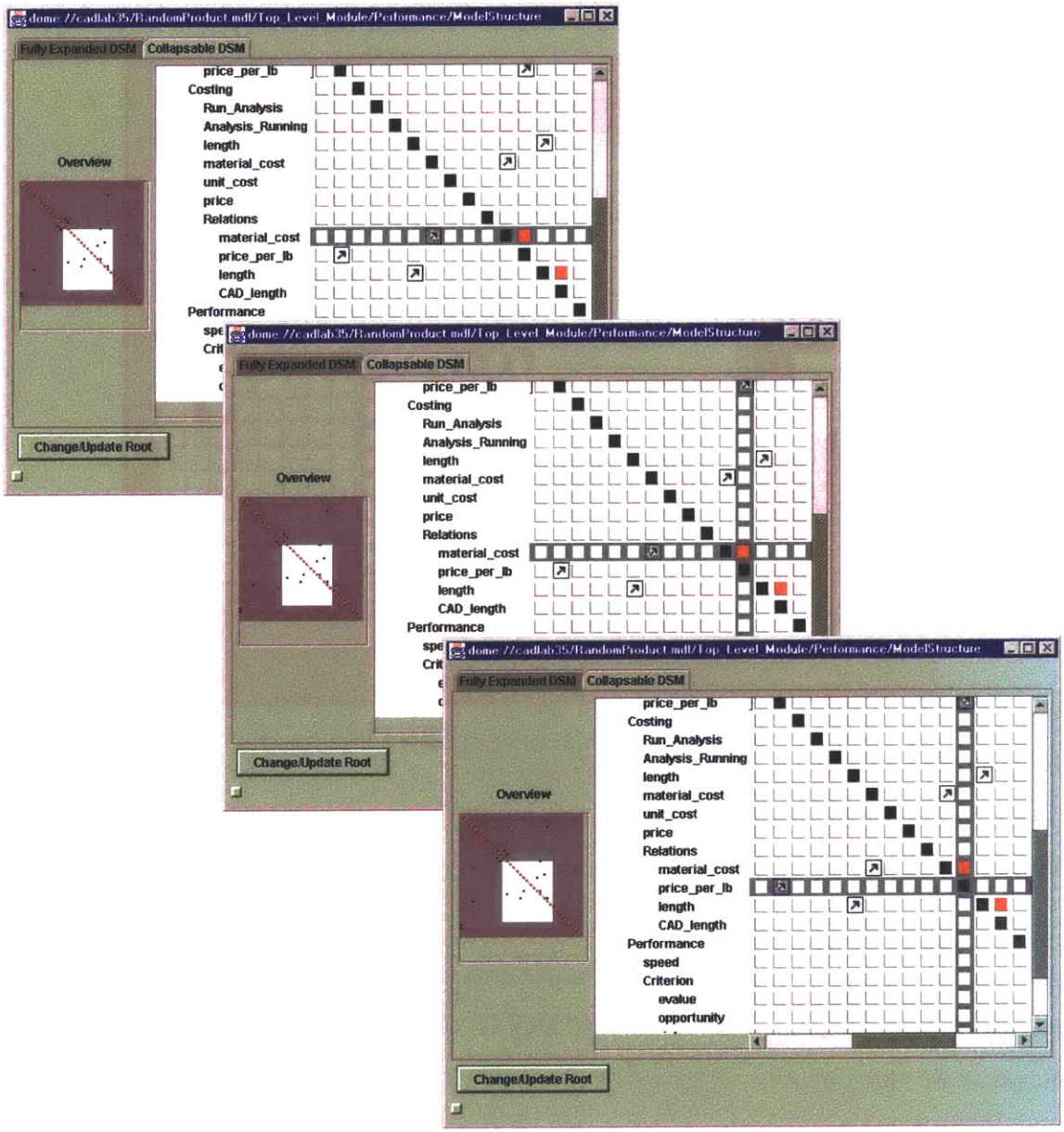


Figure 5-5: Tracing a relation dependency in an MSA matrix

As shown in the figure above, the blue cross-hairs that appear when a square is selected can be used to trace a dependency to its source through the black square along the diagonal. To find what objects a given object affects, the user reads down the given object column. Again, the blue cross-hairs can be used to trace dependencies by selecting the black diagonal square of the desired object.

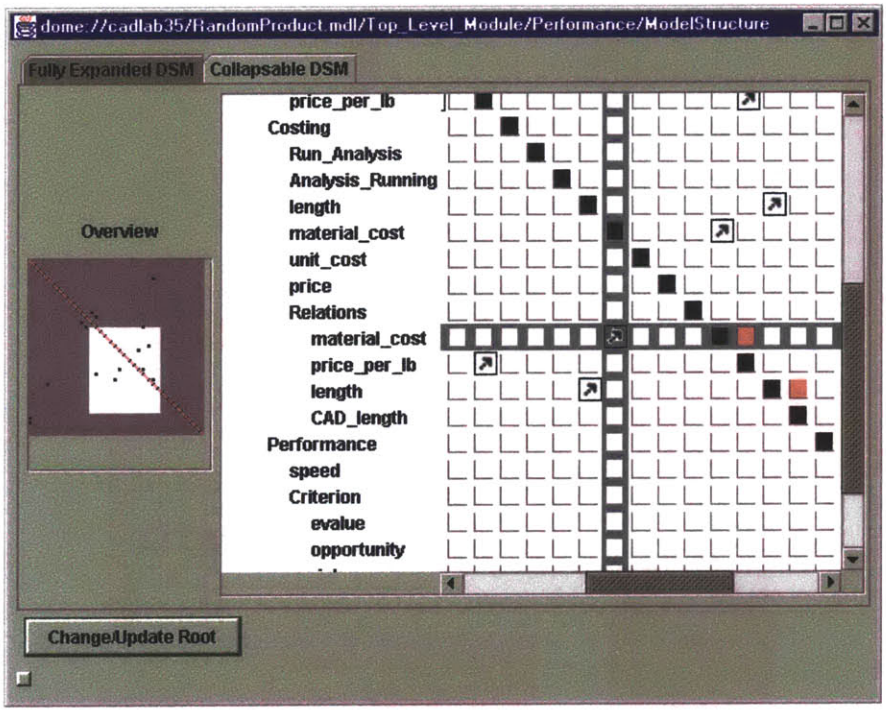


Figure 5-6: Alias dependency in a MSA matrix

Here alias dependencies are seen between the "material_cost" within the "Costing" container and the "material_cost" within the "Relations" container. Alias dependencies always appear symmetrically in a MSA because a change in either module of an alias set will drive the other module. However, because an alias object can be created in a DOME model without being mapped to another DOME object, special alias dependencies can appear non-symmetrically, as shown by the yellow squares in Figure 5-7.

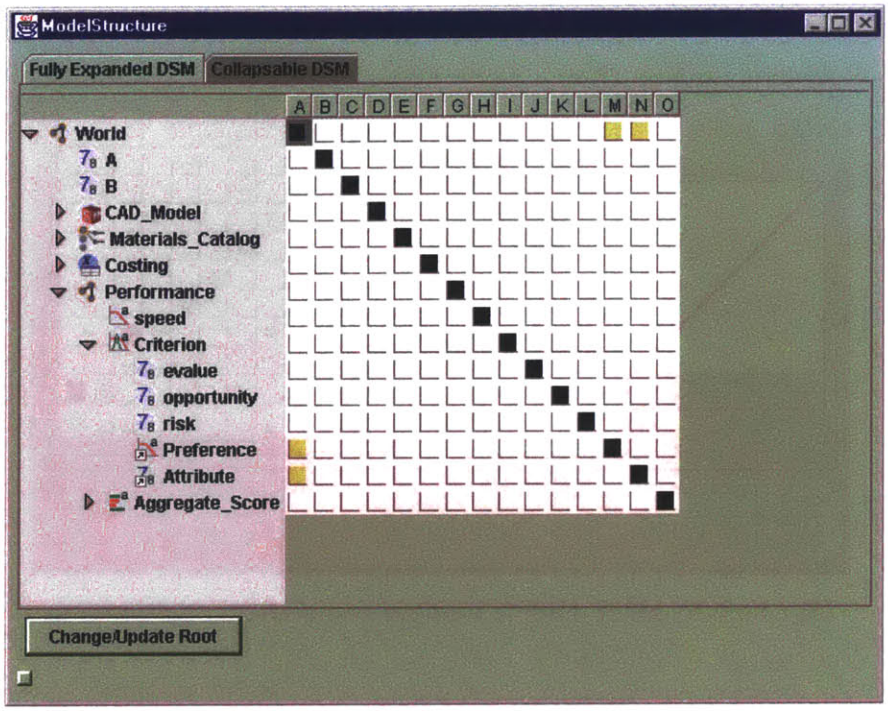


Figure 5-7: Disconnected alias dependency in a MSA matrix

Because the Criterion module objects "Preference" and "Attribute" are required to alias information from other DOME objects in order to show the Criterion evaluation, yellow disconnected alias squares are shown.

A Model Structure Analysis can be added at any level of a DOME model and can be used to analyze any scope. Using the example of the RandomProduct.mdl, if a user is concerned only with the details of the cost model and not with the structure of the entire Random Product model, he/she could add a MSA within and set the root to be the "Costing" container module as shown in Figure 5-8.

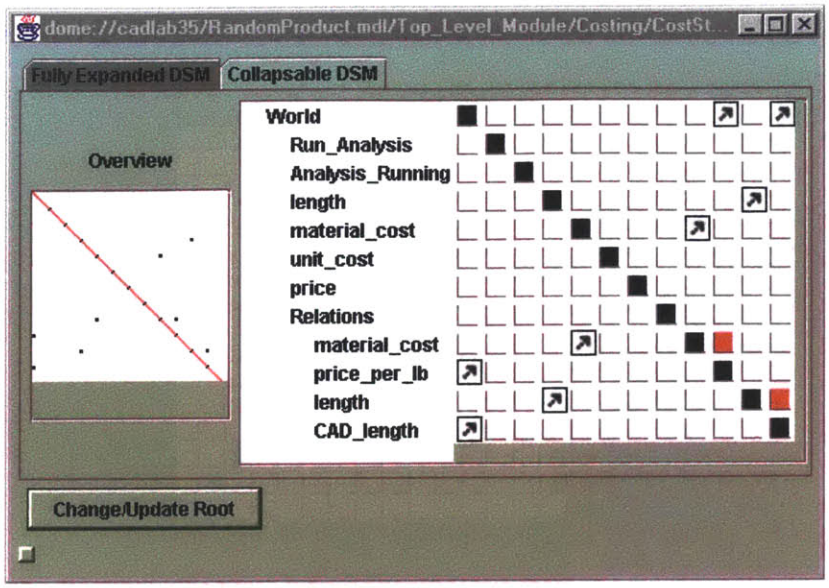


Figure 5-8: MSA used to analyze a specific model portion

Because information for the "price_per_lb" and "CAD_length" modules come from other model containers at the same level as the "Costing" container, their alias dependencies are shown mapped to "World." "World" is used to denote anything outside of the analyzed container's scope.

5.2 Use of MSA for System Design

As explained in the previous section, Model Structure Analyses can be built quickly and easily and can be added in multiple places within DOME models. Little time and overhead is required to generate an analysis, and progress in the product development process does not have to be hindered. Moreover, an MSA can be created and used by any DOME user at any time both to navigate through the DOME model and to understand the general product development system structure.

What is shown in the MSA DSM is the actual system structure and process if DOME is being used to develop the product. The information included in the resulting DSM represents the most accurate and useful information to know about the product development process. The problem of getting incorrect or superfluous information from interviews or of creating an obsolete view of the product development organization is

mitigated. Also, because DOME models integrate heterogeneous data types and various product development tasks, a single DSM can be used to capture a diverse range of system information.

MSAs can be used to get system information repeatedly and throughout the building and use of an integrated DOME product model and thus throughout the product development process. As the product development system emerges, the MSA can be used to understand the system and get immediate feedback when changes are made to the system. Because of the ease of its creation and use, MSAs can be used to help guide system structure decisions during the growing and changing of a product development system. If a poor decision with regards to the system structure is made, use of an MSA can reveal this, and the model can be altered appropriately. Additionally, the MSA may be used identify requested services that have not yet been fulfilled in the form of disconnected aliases.

Because of the ease of building and changing DOME models, DOME users can afford to explore different system structure options at little cost. The product development system structure and overall process can actually be designed.

Additionally, because the MSA can be saved in its present state, it can be used to take snapshots of the DOME model structure to monitor its evolution (Abrahamson, Wallace, Senin, Sferro 1999). Keeping records of past system states may be useful both during and after the product's development.

6 INDUSTRIAL APPLICATION OF MARKETPLACE ENVIRONMENT AND SYSTEM STRUCTURE ANALYSIS TOOL

This chapter details a project completed during the summer of 1999 to pilot a working DOME system at a large automotive company. The purpose of the project was to test the real use of DOME in an industrial environment to determine its distributed computing capabilities and its ability to operate on a large scale, as well as to discover any unforeseen issues. The resulting DOME integrated product models were then used for testing the Model Structure Analysis tool.

6.1 MGS Development Process at Automotive Manufacturer

The product development process focused on was the design of a moveable glass subsystem (MGS), the window glass lifting and sealing system contained in an automobile front door as shown in Figure 6-1. The MGS primarily includes window glass, door sheet metal, multiple seals between the glass and the sheet metal, a mechanism for raising and lowering the glass, and a regulator that drives the mechanism. To limit the scope of the project and focus on the most problematic area, the project centered on the interaction between the OEM and seal supplier. The design of the molded corner (seen at the upper right corner of the window in Figure 6-1) and major sheet metal and seals surrounding it became the topic of study.

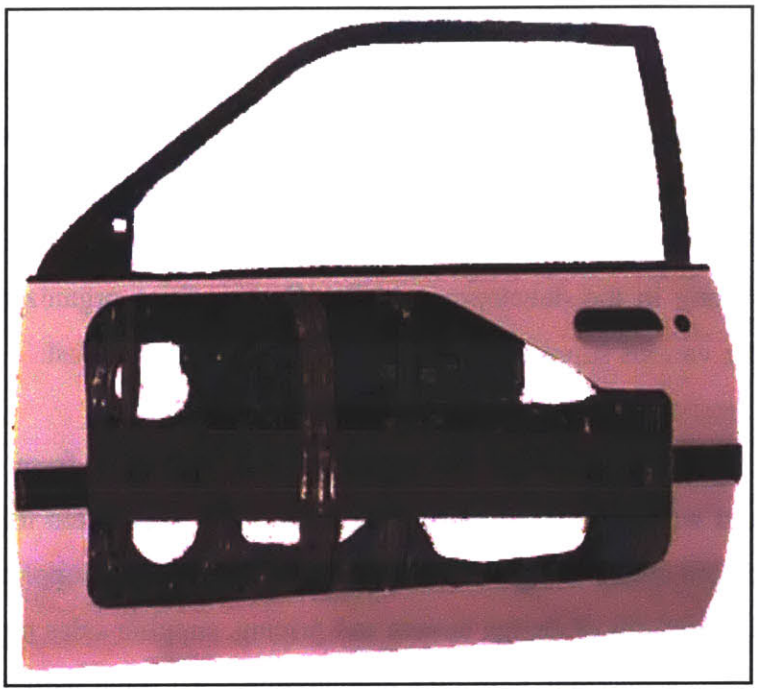


Figure 6-1: Automobile door with exposed Moveable Glass Subsystem

The MGS development process involved the coordination of many distributed parties including the automobile manufacturer as the OEM and its partnering suppliers. Due to the history of difficulty in developing reliable window systems, MGS was identified as an area that could potentially benefit the implementation DOME. The general process and set of 3rd party models had been reused for the development of past moveable glass subsystems and would be used on future doors. Thus, the OEM and suppliers were interested in setting up a functioning and accurate DOME system. Additionally, the complexity and scale of the MGS development process was identified as a good test-bed for DOME issues.

6.1.1 MGS seal supplier selection and early development process

Supplier selection for the MGS followed process mandated by the OEM supplier selection guidelines, a predefined sequential process. OEM Commodities group maintained a list of suppliers. For a new supplier to be considered for sourcing, a formal recommendation had to be made to Commodities first, then Commodities and Engineering would evaluate the supplier.

In order for Engineering to work with a supplier on the development a given MGS, the engineering team would have to give the Commodities group a document outlining its requirements for the supplier at the beginning of the vehicle program. The team was thus forced to give specific requirements early on even though those requirements might change further along in the development. The Commodities organization then would negotiate with the various suppliers and come up with an approved list.

To select the final supplier from the approved list, a person from the Purchasing department would work with the Engineering team. The augmented team would select the supplier and then create an agreement with design targets that supplier would have to meet. During negotiations of design targets and pricing, supplier sales representative also have to do internal discussions with own engineering. All three parties of Purchasing, Engineering, and Supplier representative were required to sign the agreement. This step in the process forced the process to converge on a design configuration and to freeze before continuing with the development of the MGS. Much discussion was required back and forth wasting valuable development time. Moreover, the final suppliers were not always selected according to the most correct, final design criteria because supplier selection typically occurred or one to two years before the sheet metal design was finished.

According to the Purchasing organization's rating system, only seal Supplier A was of preferred status, and historically that supplier was most often used. The Engineering group wanted to be able to consider another supplier, but it was too difficult to seriously consider one, and there was not enough time or human resources within the development process to do so. Although for each new vehicle program the door development teams would go through the process of considering different suppliers, Supplier A would win most of the business. According to some interviewed engineers, the exercise of considering other sourcing options was not serious. At the same time, the engineers recognized that other seal suppliers might be offering very competitive services. Some

engineers felt discouraged by the formal supplier selection process and that they were not using the best service providers for their products.

6.1.2 MGS general design process

The design of the MGS was centered at the OEM vehicle engineering group. The engineering group directly controlled the design of the door sheet metal around the glass and also was ultimately responsible to the rest of the company for the performance of the entire Moveable Glass Subsystem. The production of the window glass was outsourced, but its specifications were determined by the OEM. The window-lifting mechanism and regulator were also outsourced to suppliers, but the OEM predicts the window velocity performance based on the characteristics of the standard components. Both the design and production of the seals were outsourced to a seal supplier.

Typically the design of the MGS begins with the OEM engineering group. With the rest of the vehicle door team the OEM engineering group designs the sheet metal. The engineering group then sends the sheet metal files to the seal supplier design group. The seal design group uses the sheet metal files to design each of the seals. The seal design is then sent to the FEA expert who takes a couple of days to perform the appropriate analyses. If the results of the analyses are not satisfactory, the seal designers must make changes with their designs and iterate with the FEA expert. If a working design cannot be found, the seal designers may try to negotiate changes in the sheet metal parameters with the OEM engineering group.

Information from the seal designs is given to the sales group that determines seal production costs and compares them to the agreements with the OEM. If the cost and other requirements of OEM are not met, the sales group may iterate its analysis with the design of the seals which, in turn, may further iterate with the seal FEA. If the seal designs still cannot meet the OEM's requirements, the Sales group may need to renegotiate terms with the OEM Purchasing group.

Parameters from the sheet metal design are used to predict window velocity performance and stall force. These measurements are then checked against Moveable Glass

Subsystem standards, and design is iterated until the standards are met. Sheet metal information also is used in Purchasing cost models. Again, if the designs do not satisfy the cost requirements then further iteration must occur.

Finally, regardless of whether satisfactory designs for all parts of the MGS have been settled on, changes in the rest of the vehicle door may force changes in the door sheet metal design at any time causing further iteration.

Figure 6-2 schematically illustrates the information flows between the parties involved in the MGS development, as were previously described.

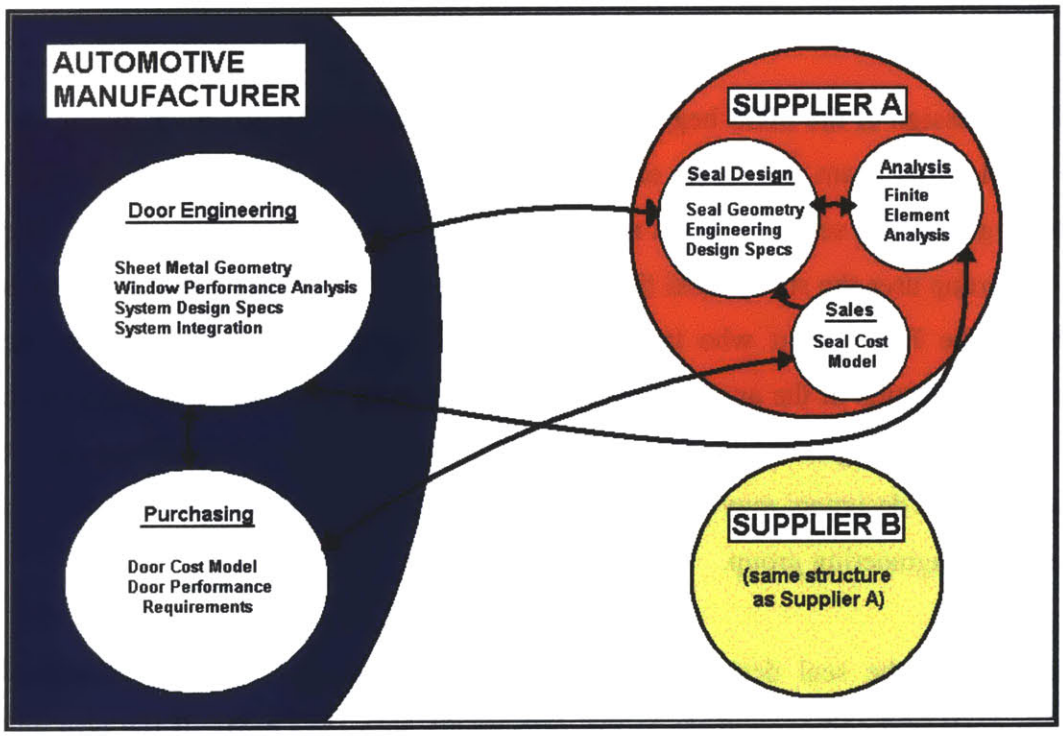


Figure 6-2: Conceptual view of MGS development process

The figure shows the MGS process as it normally occurs, with the OEM doing business with Supplier A. Supplier B, who had not been sourced before, remained at potential supplier status. However, Supplier B was able to provide the same services as Supplier A, and its internal structure and information flows were effectively the same as those of Supplier A.

Finally, because the time and effort to explore design solutions was too great, serious design changes were rarely made. Due to product development schedule and resource constraints, virtually the same MGS designs were used across the different vehicles. Only minor alterations without thorough analysis were ever made, according to MGS experts.

6.1.3 MGS design process results

The complexity of the MGS development process bred many challenges and problems, especially regarding system-level issues and other integration points. Incomplete knowledge or misdirected communications often led to confusion (Whitney, Dong, Judson, and Mascoli 1999). In particular, the difficulty of coordinating design with suppliers resulted in poorly sealing window systems due to incompatibilities between the glass, sheet metal, and seals. Because problems often were not identified until early vehicle build stages, the seal supplier often was forced to make last-minute design changes and manufacture new seals. The seal supplier noted that in spite of official price negotiations through the Purchasing organization, significant unofficial practice occurred. Consequently, the supplier's last-minute work resulted in added costs for them without added payment. Sometimes problems were not identified until vehicle production, in which case patch seals had to be installed along with the regular seals to force a good fit. Consequently, significant cost are incurred annually due to related warranty issues.

In short, pressure to finish design according to the aggressive timeline of the vehicle combined with limited resources meant that generally not all MGS design specifications could be met within the given time frame.

6.2 Pilot Project Set-up

The DOME MGS pilot project was set up to represent the actual MGS development process as accurately as possible. Following the MGS development process as illustrated in Figure 6-3, the third party models used by the MGS parties as well as newly created

models were integrated in the DOME environment. The third party model software applications are shown as they existed on the DOME server machines in Figure 6-3.

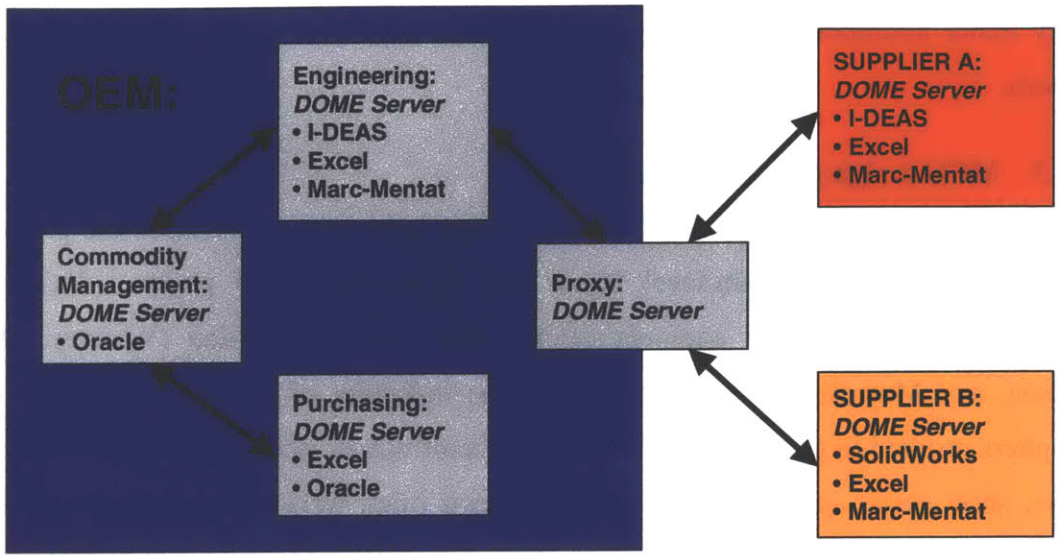


Figure 6-3: MGS Server and third party software set-up

As shown above, a DOME model server resided at each of the major participants' sites. In addition, a special proxy DOME server was also needed to mediate interactions across the OEM firewall. Four existing models, three partially completed models, and twelve new models were connected across different companies and over the OEM firewall. The third party models included eight CAD models, five FEA models, five cost spreadsheets, and one design rules database. Further details of the pilot project can be seen in Appendix B.

6.3 Pilot Project Work

The bulk of the work to complete the MGS pilot project took place over three months. Two MIT graduate students relocated to the OEM site to work directly with the OEM and suppliers. Three people in the CADlab supported the project from MIT and spent time at OEM site as well. At the OEM a special team was formed to assist the MIT team and

check the resulting DOME models for accuracy of representation. In addition, representatives from each of the MGS product development groups were either involved in third party model content creation or consulted for process knowledge and representation accuracy. On-line resources at the OEM were also consulted for some guidance.

Because some people involved in the sheet metal and seal geometric design process were not sure of all the information they required to design their parts, they were forced to designate some dimensions as the pertinent dimensions for their service interfaces. Additionally, the design goals were not always clear and had to be defined. Thus, some new portions of the geometric design interfaces were created through the project and imposed on the existing process.

Further details of the models completed and people involved in the MGS pilot project are contained in Appendix B.

6.4 Testing of the Model Structure Analysis Tool

The MGS pilot project was used to evaluate the Model Structure Analysis tool for accuracy and usefulness. The case study allowed the MSA applied to parts of the MGS integrated DOME model to be compared with DSMs created a year earlier on the vehicle door development process, including the MGS, by Qi Dong (Dong 1999).

6.4.1 Benchmark for MGS design process analysis

6.4.1.1 Interview method DSM analysis effort

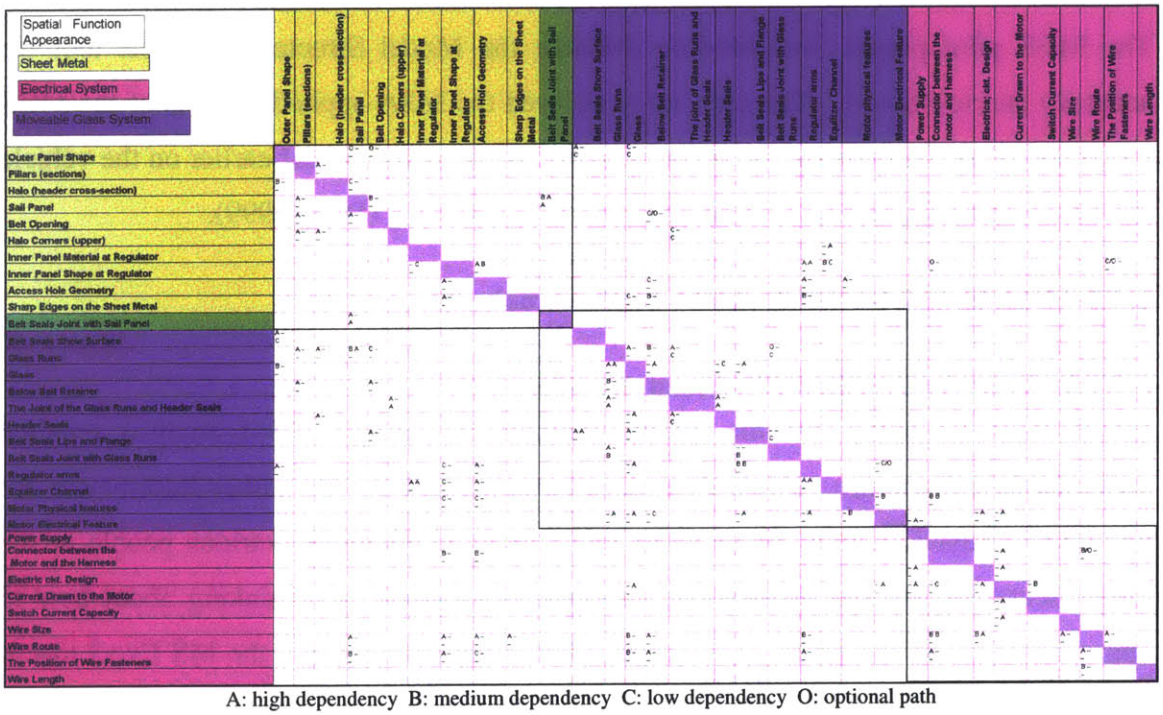
Firs it must be noted that the efforts recorded for Dong's DSM analysis entailed understanding the entire vehicle door development process. To gather information for her DSM analysis, it took Dong about one month of talking to various people at the OEM, searching for and reading documentation by herself, and thinking about how to appropriately form the DSM. The actual information entering into the MS Excel macro building DSM and analysis of the information took approximately one additional week.

To create the DSMs, Dong executed a lengthy process involving many steps: defining the system and scope to be analyzed, identifying the system elements, defining the interactions between elements, redefining the system elements, categorizing information flow contents, and assigning dependency values to relationships. Dong commented that from the interviewing and information gathering process there were often holes and inconsistencies in the information that she would have to somehow resolve through further interviews or other investigation.

Due to the way people understand system interactions and complex problems, Dong had to approach the door design process in multiple ways to elicit the correct interactions from the interviewees and create comprehensive DSMs. The different DSM types included a general design process DSM, a packaging spatial relationship DSM, a functional relationship DSM, and an appearance relationship DSM. From these multiple views of the system she synthesized a DSM of the whole process as well.

6.4.1.2 DSM results

The whole vehicle door design process DSM is shown in Figure 6-4.



A: high dependency B: medium dependency C: low dependency O: optional path

Figure 6-4: Vehicle door design process (Dong 1999)

The degree of the dependencies shown are specified as high, medium, low, or optional. Most recorded dependencies are high. Many of the dependencies are fairly scattered and do not appear to be particularly grouped. The analyzed system elements are grouped into three door areas with the Moveable Glass Subsystem portion, the primary area of interest, shown in the center in blue. The elements modeled in this DSM that correspond to the elements in the MSA DSM of section 6.4.2.2 are only "Pillars (sections)" and "Halo (header cross-section)" in the Sheet Metal area and "Glass Runs," "The Joint of the Glass Runs and Header Seals," and "Header Seals" in the MGS area. As expected from the development process, many dependencies exist between these elements.

This DSM represents the vehicle door development process as it was interpreted from interviews of development process participants during the summer of 1998. It is only a snapshot of the process at that time, and if more information was to be included or other changes made it would be difficult to alter the DSM. Moreover, if the development process itself changed this DSM would become obsolete, and a new one would have to be made.

6.4.2 Pilot project Model Structure Analysis

Because the system analyzed in Dong's DSM most closely matches that of the OEM engineering model, the Model Structure Analysis of the engineering model is compared to it here. The MSAs for the other MGS DOME models are contained in Appendix C.

6.4.2.1 Model Structure Analysis effort

The MSA was constructed as described in section 5.1.1. Once the DOME Engineering model was started and the MSA object added, the building of the MSA took approximately 30 seconds.

6.4.2.2 MSA results

The Model Structure Analysis for the OEM engineering DOME model is shown below. One MSA was applied to the entire model so the whole structure could be seen. The DSM window to the right shows only a tiny portion of the entire DSM (as denoted by the small white rectangle in the Overview). The Overview window to the left provides a convenient snapshot of the whole structure.

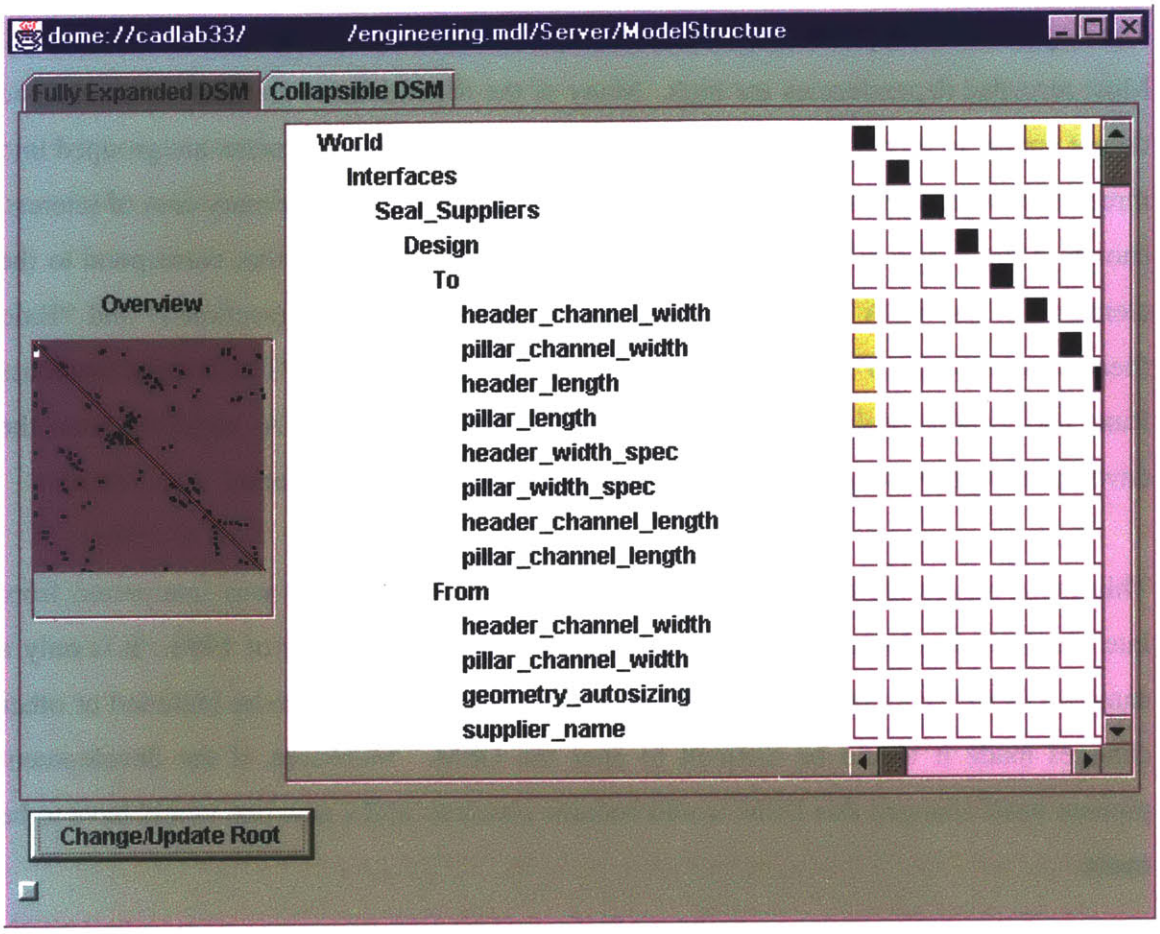


Figure 6-5: Model Structure Analysis for DOME Engineering model

Many of the dependency marks shown in the MSA DSM are due to service-objects aliasing each other. The current implementation of the research software requires that aliases to or the service-objects themselves be in the same container in order to write relation-objects between them. Thus, the high number of aliases are necessary for the service-objects to be able to pass information between them. The many alias dependencies here have created patterns in DSM that are mostly symmetric about the diagonal.

This DSM represents the system in its state at the moment the root of the analysis was set, so the structure shown reflects the services that were connected to at that moment, such as the particular supplier that was being sourced. In this way, the MSA is generally up-to-date and is a living document.

6.4.3 Comparison of Model Structure Analysis results to benchmark

It is clear that the system elements represented in MSA DSMs were very different from those identified in Dong's DSMs. The scope analyzed by the two approaches differs as well. Dong's DSMs cover mainly the design process within the OEM Engineering group but involve more components of the vehicle door, while the MSA DSMs include the interactions between the Engineering group, Purchasing, Commodities, and seal suppliers but for fewer door components in finer detail. This fundamental difference between the two DSM approaches makes them extremely difficult to compare. However, the existence of dependencies between the relevant elements in Dong's DSM are consistent with the dependencies shown in the DOME engineering Model Structure Analysis.

It is unclear whether one approach's results were more correct than another's. Dong's approach was based on the knowledge and interpretation of people involved in the MGS development process, while the MSA was based on the parameter interactions and computer models used in the design and evaluation process. With the DSM interviewing approach, error could have been introduced in the information gathering or interpreting processes. The MSA results, however, were directly extracted from the MGS DOME models so that error was not introduced into the extraction process itself. Yet, the MSA results were only as accurate as long as the participants exchange services and define relationships with the DOME environment.

It is clear that the effort involved in the process information gathering and DSM generation was significantly less using the MSA on the existing DOME model than interviewing development process participants and manually entering the information into the Excel DSM program.

6.5 Case Study Results

6.5.1 Accuracy of the MGS integrated DOME model

The setting up of the DOME system for the MGS development process caused some of the process to change to suit the service marketplace environment. It must be considered

to what extent the DOME system representation of the process was different from the actual process and how this would affect the actual process if the DOME system is used.

The representation of the design tasks in terms of ownership and responsibility by different parties at the highest level was formed based on interviews of the product development participants. At the DOME service-object level, the grouping and encapsulation of objects in different containers was influenced somewhat by the functionality of the research software at the time and somewhat by the primary DOME model builder.

The creation of the service interfaces and the service-objects that made up the DOME models were foremost driven by the connecting and use of the third party models. However, to fill in the rest of the gaps, the people involved in the process were interviewed and in some cases forced to clarify or even define what information pieces they needed from and gave to others. In some areas, it was not known exactly what the criteria were for good design, how to measure the criteria, or what the design targets or acceptable parameter boundaries were.

Although the transition to the DOME system may have forced people to articulate new portions of the development process, the resulting parameter interactions may more fully have reflected what information the people really needed. Furthermore, it is reasonable to assert that clarification of such information should be necessary in any good design process. Thus, although the setting up of the DOME system may not have entirely accurately represented the actual process, the exercise perhaps helped improve the process.

Finally, if the DOME system is used in the development process, then it must include the parameters that actually are relevant. The degree to which it is used can only improve the DOME model accuracy. In short, if used, the DOME system becomes the development process.

6.5.2 DOME as an emergent system environment

According to the characteristics developed in section 3.3 and summarized in Table 3-1, the DOME software provided the MGS development process with an emergent system environment in many ways. Clearly the resources and knowledge of the system were distributed among the many system elements. The third party models, DOME integrated models, and other design and proprietary information all were located on different DOME servers at different parties' sites.

Moreover, design activity and control were distributed among the development participants. All of the system transactions of making services available, requesting services, and connecting to services were accomplished by the individual elements. Minimal intervention by an overseeing power was needed to create the MGS DOME system, and the resulting MGS development system did not require the OEM Engineering or Purchasing groups to supervise the majority of the Suppliers' activity.

The open architecture of DOME introduced a blank slate onto which the MGS development process could be accurately represented and then allowed the development process to grow and change on its own. On a functional level, use of the DOME software did not force the MGS development system to conform to any specific process structures.

The low transaction costs of subscribing to different services and ease of getting feedback within the DOME system enabled the OEM to consider the new option of Supplier B. Furthermore, the use of DOME forced the competing suppliers to create the same interfaces to respond to the OEM service requests. Consequently, the OEM was less impeded by the cumbersome supplier selection process, and the ground to compete for the OEM's business became more fair. Once Supplier B was considered for business, it was predicted that Supplier B could likely yield higher quality, more thorough, and more timely services than Supplier A. Thus, DOME made it possible for a new party and more suppliers in the future to compete in the MGS product development system. At the same time, DOME enabled the OEM and the MGS product development system to select the best options and made evolution to new system structure easy.

Finally, throughout the process DOME made an understanding of how entire system functioned and feedback about the consequences of design decisions available to the participants. Representations of the integrated model structures were generated by the Model Structure Analysis tool.

6.5.3 Evaluation of the DOME Model Structure Analysis tool

The Model Structure Analysis tool introduced an easy, rapid way to understand the interactions between the system elements in the DOME MGS integrated product models. Compared to the traditional method of generating DSMs, the MSA reduces process extraction time and effort. Also, the MSA method of DSM creation does not risk losing detail or misrepresenting the product development process as does constructing DSMs through interviews. Because the MSA extracts the process structure directly from the DOME model, it has the potential to contain greater information quality and depth.

The MSA provides a compact way of looking at the entire system at once. Using the interviewing manual method, Dong had to create four DSMs to represent all of the system interactions, while the MSA combined all types of elements and interactions into the same DSM. Depending on the intended application, using the MSA could be advantageous or disadvantageous. However, the MSA allows specific portions of the product development system to be analyzed and different levels of encapsulation to be viewed. Thus the level of detail seen can be controlled to avoid being overwhelmed by too much information.

One important thing to note is that the traditional DSMs can contain information about the strength of the dependencies between the system elements. Currently the DOME MSA plug-in does not discern dependency strength but only shows direct and alias relationships. However, one can envision using sensitivity analysis techniques to automatically generate this information.

The Model Structure Analysis tool can be used to gather system information at anytime during the product development process. Thus, it can aid the design of the product

development organization and process by giving feedback on its structure. The MSA can also aid in decision support when choices affecting the system structure are made.

The DOME software must have some way of analyzing and monitoring the system structure if the product development system is allowed to evolve on its own. In this way the MSA is an integral part of an emergent product development process environment. Additionally, the MSA can produce useful feedback on rigid, non-emergent product development systems built in DOME. Whether the system is built in a top-down manner or is allowed to emerge via distributed activities, the MSA serves as a critical tool in product development system analysis and management.

6.6 Changing from a Traditional to an Emergent Product Development System

The exercise of implementing the DOME model for the Moveable Glass Subsystem generated many findings about transforming a traditional product development system into an emergent one.

A number of difficulties were encountered in trying to set up the DOME system. To begin with, many of the people involved in the MGS development process did not fully know how they needed to directly interact with others. Many did not know what exact information they needed to do their jobs and what information they produced that was pertinent. This lack of clarity made the defining of DOME service interfaces challenging, one must also wonder how this lack of understanding impacted the preexisting development activity.

Additionally, because the service marketplace environment brings together competing and bargaining parties, competitors must learn initially to trust the software and those that build the DOME models. Although less risk is actually involved than is perceived, trust is needed to obtain the cooperation to implement the system.

Finally, in order to motivate the participants to make the DOME system work, some sort of incentive system should be put in place. The right incentives are particularly important for those whose advantage positions may be compromised such as supplier being threatened with losing stronghold on business or managers losing control.

From these experiences a list of challenges were identified that could prevent the building of a successful DOME system. These are summarized in Table 6-1.

Table 6-1: Barriers to implementing a DOME system

- People do not know where they get information from
- People do not understand who uses their information
- People do not know what information is useful to them
- Correct use of the models is unclear
- Design specifications and goals are unclear
- Methods for judging how good a design is are unclear
- Processes for making decisions are unclear
- Old software, old processes, tradition
- Wrong or no incentives, no visible immediate payback
- Mistrusting participants

These encountered difficulties gave rise to a set of suggested requirements needed to effectively implement DOME in a product development project. Table 6-2 compiles these requirements under general categories related to technological capability of the product development participants, company cultures, and clarity of the existing process.

Table 6-2: Attributes of the product development organization needed for a successful DOME project implementation

Technological readiness	• Updated software applications • WWW enabled, firewalls (if any) understood • Work force that can integrate new skills
Right organizational culture, climate	• Willingness to learn • Desire to improve development capabilities • Minimal trust between participants • Supportive management • Willingness to change roles and responsibilities
Appropriate application	• Available functional third party models • Engineering issues that can be solved • Good local process knowledge • Clear design targets and ways of measuring them

Once these requirements are met, the resulting product development system should have certain characteristics to enable it to be an efficient emergent system. These characteristics are summarized in Table 6-3.

Table 6-3: Desired characteristics of an emergent product development system

- Many available design alternatives
- Participation in system as easy as possible and possible for all
- Enough flexibility and openness to accepting different solutions present in the system that new possibilities can be experimented with, and creative new solutions are allowed
- System readily understood at any point in time to make navigation of it and communication within it easy
- Structure may be minimally altered as needed to improve efficiency
- Changing the participants and structure of the product development organization and process is as easy and fluid as possible
- Comparing design alternatives fast and operationally straight forward

7 CONCLUSIONS

7.1 Summary

In this thesis the notion of product development as an emergent system is explored. Traditional sequential approaches to product development tend to follow rigid process structures that historically may have been needed to overcome communication and complexity barriers. However, in a changing environment these structures may act as hindrances. Sequential product development requires central coordination of the process and thus presents problems of inefficiency due to ineffectively used resources and long communication times. Although concurrent engineering addresses these issues, it promotes a well-defined, constant development process structure that cannot adapt to change. Moreover, due to technological or communication limitations, true concurrent engineering has not yet been fully realized in industrial practice. Selection of supply partners in product development also suffers due to the sequential, rigid nature of traditional product development.

Emergent systems such as the free economic marketplace exhibit many characteristics desired in a product development system. Namely, emergent systems with knowledgeable, independently acting elements result in the best choices for the system and a generally efficient allocation of resources. Additionally, emergent systems are able to explore new options and solutions and adapt well to changes in the environment.

The DOME framework is presented as a means of providing an emergent system environment for product development. The fluid structure and distributed nature of DOME allow the product modeled in DOME and its development organization to change, emerge, and evolve as needed. In this simulated product development service marketplace, transaction costs are kept low and service providers required to answer requests in the same format so that design options can be readily explored and competition remains fair.

A plug-in to the DOME research software was developed to extend the emergent system environment capabilities. The Model Structure Analysis tool rapidly generates a Design Structure Matrix view of the system element dependencies so that DOME users can understand the effects of design actions on others. Additionally, the MSA enables navigation and monitoring of the emerging system throughout the product's development. The MSA helps make use of the emergent system environment possible, manage structural chaos, and increase understanding for all participants in development process.

A case study with the development of a vehicle Moveable Glass Subsystem served to test the emergent DOME system and the MSA tool. The case study focused on interaction between the OEM and two of its suppliers. Use of the DOME system drastically altered the MGS development from a highly sequential and OEM-controlled process to a concurrent, distributed process. Furthermore, the OEM was able to investigate many more design alternatives by rapidly considering each of the suppliers at will. The MSA tool also rapidly yielded necessary process and other system information.

7.2 Limitations of the Approach

Use of the DOME framework to create an emergent system environment for product development presents a number of limitations. The natures of both the DOME system and product development and their interaction to some degree prevent a totally emergent environment from being realized.

First, a DOME product model is largely dependent on the computer-based models that it integrates and that are built in the DOME software itself. Although manual entry of information or other significant repeated human interaction can easily be included in the DOME product development process, it significantly slows feedback time. This, in turn, slows the time it takes to examine design alternatives which decreases the number of alternatives explored during a product's development, given development project time

constraints. The slower the rate of change and the fewer options that can be considered, the less emergent is the system.

DOME addresses design parameters and other numbers or pieces of definite information so DOME model builders must be able to approach design decisions quantitatively, having standards for good designs and knowing how to measure them in order to evaluate alternatives in DOME system. If only certain portions of the entire product design process can be built in this quantified manner, then the resulting DOME model and product development process will focus on those parts of the design.

Because DOME models build on quantifiable design activity, some methods of decision making and fuzzy design rationale are not accounted for. The design process needs to be articulated well, and the design participants need to know minimally what information they need and what information they can provide to others, or that portion of the process will not be represented in the DOME model. Consequently, the entire design and development process may not be accurately captured in the DOME environment.

Regarding product development itself, the process as practiced in industry today and the organizations and people that participate in it introduce additional limitations to an emergent system. Foremost, despite the fluidity of the supporting software models and structure, the participants themselves may not be able to change quickly and adjust appropriately to the emerging system. Or, while some or the groups may be able to move at a higher speed, the limitations of other groups may hold back the emergence of the entire system.

As a method for capturing the product development process, the Model Structure Analysis plug-in exhibits some limitations. Because the MSA directly extracts the process from DOME models, it does not capture any dependencies not represented in DOME. Less apparent influences in the development process and the relative strength of dependencies do not show in the model analysis. Finally, the model structure and

resulting analysis is inherently contingent on the model builder's organization of the model.

7.3 Future Work

The information presented in this thesis gives rise to more work to be done and many questions that remain to be answered.

7.3.1 Plug-in development

In order to better the MSA tool's analysis quality, a number of improvements should be introduced to the plug-in. To focus on the important dependencies, MSA users should be able to control whether or not alias dependencies are displayed in the DSM. On the collapsible tree view panel of the MSA, some symbol on the DSM should signify when a dependency exists inside of a collapsed container so that the user knows to expand the container. Also, when service-objects are shown aliasing a service-object outside of the MSA scope ("World"), more information should be given about where to find the object that is aliased.

Some issues with the limitations of the Model Structure Analysis tool could be addressed by making the resulting DSMs editable so that users could put in annotations or additional dependency information that is not be captured in the DOME model. If desired then the sensitivity of dependencies could also be noted. To aid in DOME model and potentially organization design, another tree view with corresponding DSM should be available in the MSA in which the grouping of the service-objects can be rearranged differently. This would allow users to experiment with how the dependency structure is affected by the element grouping.

7.3.2 Research

Significant research needs to be done regarding whether or not the DOME framework can accurately represent the product development process. Related to this, whether or not indirect decision influences and fuzzy design rationale should remain a part of the design process should be investigated.

The ability of the various product development groups to change is critical to an emergent product development environment. Thus, one should look into the time scales for change for different product development processes and how they relate to both normal organizational change and change in the DOME service marketplace. What happens when different participating groups emerge at different rates should also be questioned.

The introduction of an emergent system in which central control is minimized and the individual elements are empowered yields many new questions related to organizational and management science. How are company hierarchies are affected by the transformation to an emergent system structure in which power is more distributed? What happens if roles and responsibilities are allowed to change within a company and product development tasks are not necessarily owned by particular groups or individuals? How do the consequent roles of and need for management change as the individual entities become more autonomous and the structure becomes more fluid? How are intermediary parties like a Purchasing department affected?

Similar questions regarding the product development organization between companies need to be answered as well. If decision making becomes more distributed, how do the associated power structures change, who then will control the development process, and how will negotiations be affected? Furthermore, how will negotiation, trust, and partnerships with suppliers and strategic alliances change in such an environment of open competition?

The outcomes of evening the competitive field with a service marketplace environment must also be researched. By allowing more service providers to compete more easily over the web it is uncertain who will have the advantages and how the market will grow.

Finally, it should be explored to what degree the individual parties can become empowered before the emergent system reaches a chaotic level. Where the line between chaos and constructive emergence lies and how much control must be imposed to achieve it must be found. Moreover, how the balance may or may not change with product or organizational complexity also remains in question.

8 REFERENCES

Abrahamson, S. A., (1999), *Tools for Decomposition in an Integrated Modeling Environment*, MS Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Abrahamson, S. A., D. R. Wallace, N. Senin, and P. Sferro, (1999), "Integrated Design in a Service Marketplace", *Computer-Aided Design,* **32**(2) pp. 97-107.

Ahmadi, R. and R. Wang, (1999), "Managing Development Risk in Product Design Processes," *Operations Research*, **47**(2) pp. 235-246.

Beamon, B. M., (1998), "Supply chain design and analysis: Models and methods," *International Journal of Production Economics*, **55** pp. 281-294.

Borland, N., H. P. Kaufmann, et al, (1998), "Integrating Environmental Impact Assessment into Product Design: A collaborative modeling approach," *1998 ASME Design Engineering Technical Conferences*, September 13-16, 1998, Atlanta, GA.

Braunschvig, D., (1998), "Work Remade: An Electronic Marketplace Inside the Corporation," *The Future of the Electronic Marketplace,* Ed. Derek Leebaert, MIT Press, Cambridge, MA, pp. 177-203.

Choi, T. Y. and J. L. Hartley, (1996), "An exploration of supplier selection practices across the supply chain," *Journal of Operations Management*, **14** pp. 333-343.

Coleman, H. J., (1999), "What Enables Self-Organizing Behavior in Businesses," *Emergence*, **1**(1) pp. 33-48.

Crabtree, R. A., M. S. Fox, and N. K. Baid, (1997), "Case Studies of Coordination Activities and Problems in Collaborative Design," *Research in Engineering Design*, **9** pp. 70-84.

Dictionary.com, (2000), http://www.dictionary.com/.

Dong, Q., (1999), *Representing Information Flow and Knowledge Management in Product Design Using the Design Structure Matrix*, MS Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Emergence: A Journal of Complexity Issues in Organizations and Management, (2000), http://www.emergence.org/.

Fruchter, R., K. Reiner, L. Leifer, and G. Toye, (1998), "VisionManager: A Computer Environment for Design Evolution Capture," *Concurrent Engineering: Research and Applications*, **6**(1) pp. 71-84.

Funtowicz, S. and J. R. Ravetz, (1994), "Emergent Complex Systems," *Futures*, **26**(6) pp. 568-582.

Griffin, A., (1997), "The Effect of Project and Process Characteristics on Product Development Cycle Time," *Journal of Marketing Research*, **34** pp. 24-35.

Hameri, A. and J. Nihtila, (1997), "Distributed New Product Development Project Based on Internet and World-Wide Web: A Case Study," *Journal of Product Innovation Management*, **14** pp. 77-87.

Huang, G. Q. and K. L. Mak, (1997), "Re-engineering the product development process with 'design for X,'" *Proceedings of the Institution of Mechanical Engineers*, **212**(B), pp. 259-268.

Inayatullah, S., (1994), "Life, the universe and emergence," *Futures*, **26**(6), pp. 683-696.

Liu, S. T., personal contact, April 2000.

The MIT Design Structure Matrix—DSM Home Page, (2000), http://web.mit.edu/DSM/.

Orlikowski, W. J., (1995), "Evolving with Notes: Organizational Change around Groupware Technology," Working paper.

Pahng, F., S. Bae, and D. R. Wallace, (1998), "A web-based collaborative design modeling environment", *Proceedings of the Seventh IEEE International Workshops on Enabling Technologies*, Infrastructure for Collaborative Enterprises Conference, Stanford University, pp. 161-167.

Pindyck, R. S. and D. L. Rubinfeld, (1998), *Microeconomics*, Fourth Ed., Prentice Hall, New Jersey.

Prasad, B., R. S. Morenc, and R. M. Rangan, (1992), "Information management for concurrent engineering: research issues," *Concurrent Engineering: Research and Applications*, The Institute of Concurrent Engineering, **1** pp. 3-20.

Rogers, J. L, "Reducing Design Cycle Time and Cost through Process Resequencing," *International Conference on Engineering Design 1997*, Tampere.

Sabbaghian, N., S. Eppinger, and E. Murman, (1998), "Product Development Process Capture and Display Using Web-Based Technologies," Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, Oct. 11-14, 1998, pp. 2664-2669.

Senin, N., D. R. Wallace, and N. Borland, (2000), "Distributed Object-Based Modeling in Design Simulation Marketplace," MIT Computer-Aided Design Laboratory, in review.

Smith, R. P. and S. D. Eppinger, (1998), "Deciding between Sequential and Concurrent Tasks in Engineering Design," *Concurrent Engineering: Research and Applications*, **6**(1) pp. 15-25.

Truex, D. P., R. Baskerville, and H. Klein, (1999), "Growing Systems in Emergent Organizations," *Communications of the ACM*, **42**(8) p. 117-123.

Truex, D. P. and H. K. Klein, (1991), "A Rejection of Structure as a Basis for Information Systems Development," *Collaborative Work, Social Communications, and Information Systems,* Ed. R. K. Stamper, P. Kerola, R. Lee, K. Lytinen, North-Holland, New York, pp. 213-235.

Ulrich, K. T. and S. D. Eppinger, (2000), *Product Design and Development*, Second Ed., McGraw-Hill, Boston, MA.

Wallace, D. R., S. Abrahamson, and N. Borland, (1999), "Design Process Elicitation Through the Evaluation of Integrated Model Structures", *Proceedings of the ASME DT Conferences*, DETC/DFM-8780, Las Vegas, NV.

Wang, K. and Y. Jin, (1999), "Modeling Dependencies in Engineering Design," *Proceedings of the 1999 ASME Design Theory and Methodology Conference,* DETC/DSM-8778, Las Vegas, NV.

Wang, P. H., (1998), *Benchmarking a Collaborative, Concurrent Computer Design Tool*, BS Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Whitney, D. E., Q. Dong, J. Judson, and G. Mascoli, (1999), "Introducing Knowledge-Based Engineering into an Interconnected Product Development Process," *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, DETC/DTM-8741, Las Vegas, NV.

APPENDIX A: MODEL STRUCTURE ANALYSIS PLUG-IN CODE

This appendix includes high level information on the algorithms written in the Model Structure Analysis code.

For fully expanded DSM:

Search the model within the copied container (on the server) recursively

- For each container check the relation-objects and alias-objects inside of it
- Search through each relation, find the output, and fill in the matrix appropriately
- Search through each relation, find the inputs, and fill in the matrix appropriately
- Search through aliases and fill in matrix appropriately for aliases and their targets
- While searching through model, keep track of the level of the container and service-objects to create the flat structure of the DSM and the list of names

For collapsible DSM:

When building the tree, another version of the same tree that is in the regular DOME model must be created with the relations and MSA objects hidden relation. This is accomplished with the following steps.

- Search the model tree view within the copied container (on the client) recursively
- When a relation module or a MSA module is found add its parent container to the list of modified containers and keep track of the modified list of children so that when a child node is called the correct child will be pointed to

When the DSM is painted each time the tree is expanded or collapsed, the program must Go through each row of the visible tree

- Keep track of the visible tree nodes
- Create a new data matrix from the whole DSM including only the visible matrix elements

APPENDIX B: USE OF MGS THIRD PARTY MODELS

This appendix contains more specific information on the third party software models used in the Moveable Glass Subsystem development process.

The OEM engineering group used one three-dimensional CAD model in I-deas for the door sheet metal around upper glass. In addition, there was a three-dimensional CAD model in I-deas of the whole door configuration including the sheet metal, window-lifting mechanism and regulator, glass, and seals. Engineering also used a spreadsheet in MS Excel that calculated window raising and dropping velocity characteristics, safety stall force for glass, and characteristics of regulator. System design specifications for overall MGS performance were also taken from a company Oracle database.

Seal Supplier A had two two-dimensional CAD models in I-deas, one for each of the upper and side seals. In the analysis department a two-dimensional finite element analysis model in MARC-Mentat was used to assess the pressure on the side seal for after the glass is installed, and another two-dimensional FEA model was used to assess the pressure on the upper seal as the window is closed. Within the Sales department, a cost model in Excel calculated unit seal package price. Additionally, some internal seal design specifications were built directly into the DOME Supplier A model as design criteria for the supplier.

At seal Supplier B, basically the same models resided plus a couple of additional ones. The two two-dimensional seal CAD models for the same parts as Supplier A were built in SolidWorks. Improving upon the capabilities of Supplier A, Supplier B introduced a three-dimensional CAD model in SolidWorks of the problematic molded corner seal. Supplier B provided virtually the same two two-dimensional FEA models in MARC-Mentat, but in addition Supplier B introduced a three-dimensional FEA model in MARC-Mentat to assess sliding friction on the side seal. For seal costing, Supplier B used two cost models in Excel to calculate the price of each of the main two seals.

At the OEM Purchasing department a cost model for the entire door resided .Supplier requirements and control of the currently sourced supplier also were kept track of by Purchasing.

Table B-1 summarizes all of the previously discussed information.

Table B-1: Third party models integrated in DOME MGS Pilot Project

Model	Software	Description	Location
MGS Modeler	I-deas	Parametric CAD model of door	OEM Engineering
Velocity/Stall Force Model	MS Excel	Calculates stall force, glass velocity, and regulator parameters	OEM Engineering
Sheet Metal Geometry (3-D)	I-deas	Parametric CAD model	OEM Engineering
MGS System Specs	Oracle	Contains rules and specifications for whole MGS	OEM Engineering
Door Cost Model	MS Excel	Calculates unit price based on door and window dimensions	OEM Purchasing
Header Seal Geometry (2-D)	I-deas	Parametric CAD model	Supplier A
Pillar Seal Geometry (2-D)	I-deas	Parametric CAD model	Supplier A
Header Seal FEA (2-D)	MARC-Mentat	Analyzes pressure on glass as window is closed	Supplier A
Pillar Seal FEA (2-D)	MARC-Mentat	Analyzes pressure on installed glass	Supplier A
Seal Package Cost Model	MS Excel	Calculates unit price of seal package	Supplier A
Header Seal Geometry (2-D)	SolidWorks	Parametric CAD model	Supplier B

Pillar Seal Geometry (2-D)	SolidWorks	Parametric CAD model	Supplier B
Corner Seal Geometry (3-D)	SolidWorks	Parametric CAD model	Supplier B
Header Seal FEA (2-D)	MARC-Mentat	Analyzes pressure on glass as window is closed	Supplier B
Pillar Seal FEA (2-D)	MARC-Mentat	Analyzes pressure on installed glass	Supplier B
Pillar Seal FEA (3-D)	MARC-Mentat	Analyzes glass sliding friction	Supplier B
Header Cost Model	MS Excel	Calculates unit price of header	Supplier B
Pillar Cost Model	MS Excel	Calculates unit price of pillar	Supplier B

APPENDIX C: STRUCTURAL ANALYSES OF MGS DOME MODELS

This appendix includes Model Structure Analyses of the rest of the integrated DOME model files that make up the MGS integrated product model. Due to the size of the MSAs, they are not shown in their entirety here. The Overviews to the left of the MSA screen captures show the patterns in the DSMs most effectively in snapshots. First the other models inside the OEM firewall are exhibited, then are those that deal with the suppliers. The Commodities model structure is displayed below.

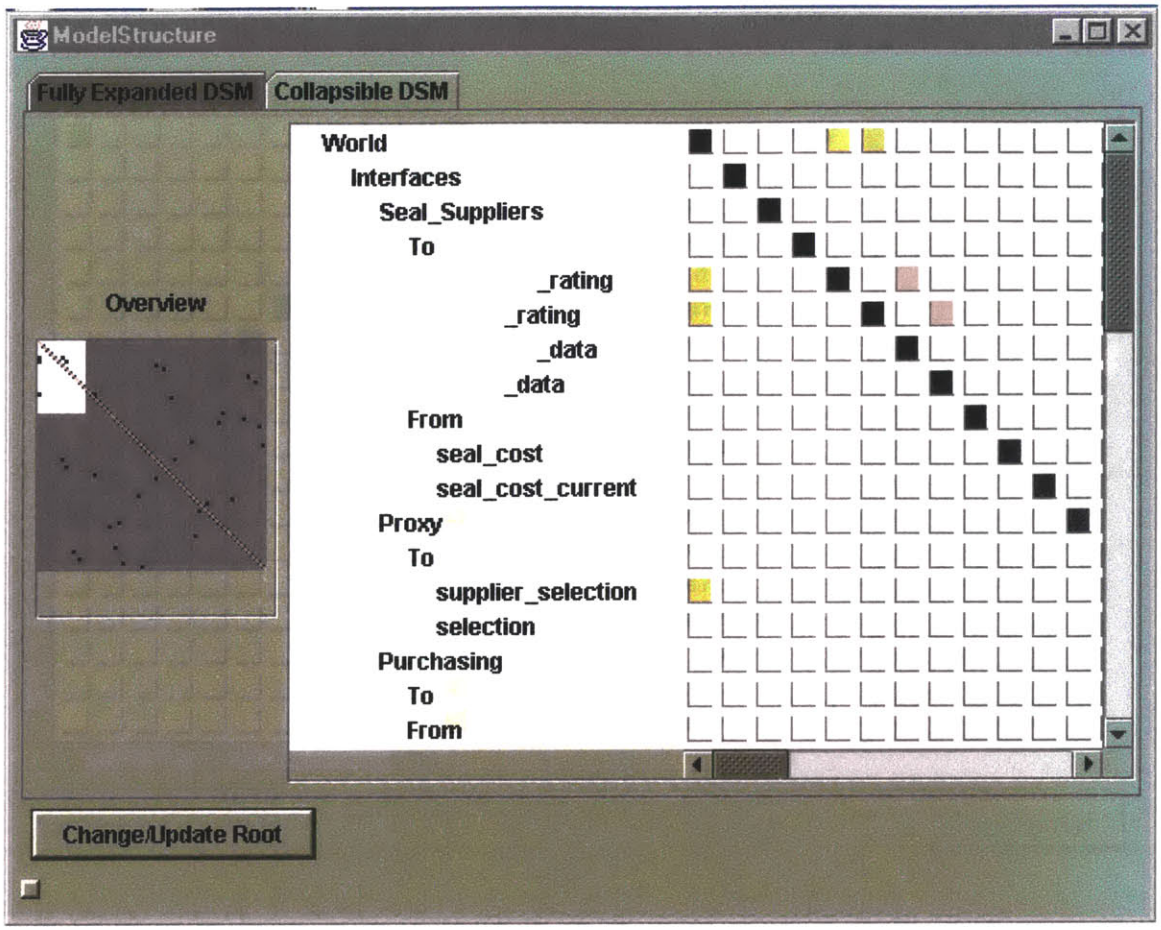


Figure A-1: Commodities Model Structure Analysis

In both the Commodities model DSM and the Purchasing model shown in Figure A-2, most of the dependencies shown are symmetric due to alias-objects needed to write relationships between the model's service-objects. The few non-symmetric interactions are shown at the upper left corner (seen more easily in the Overviews), representing information exchange with suppliers.

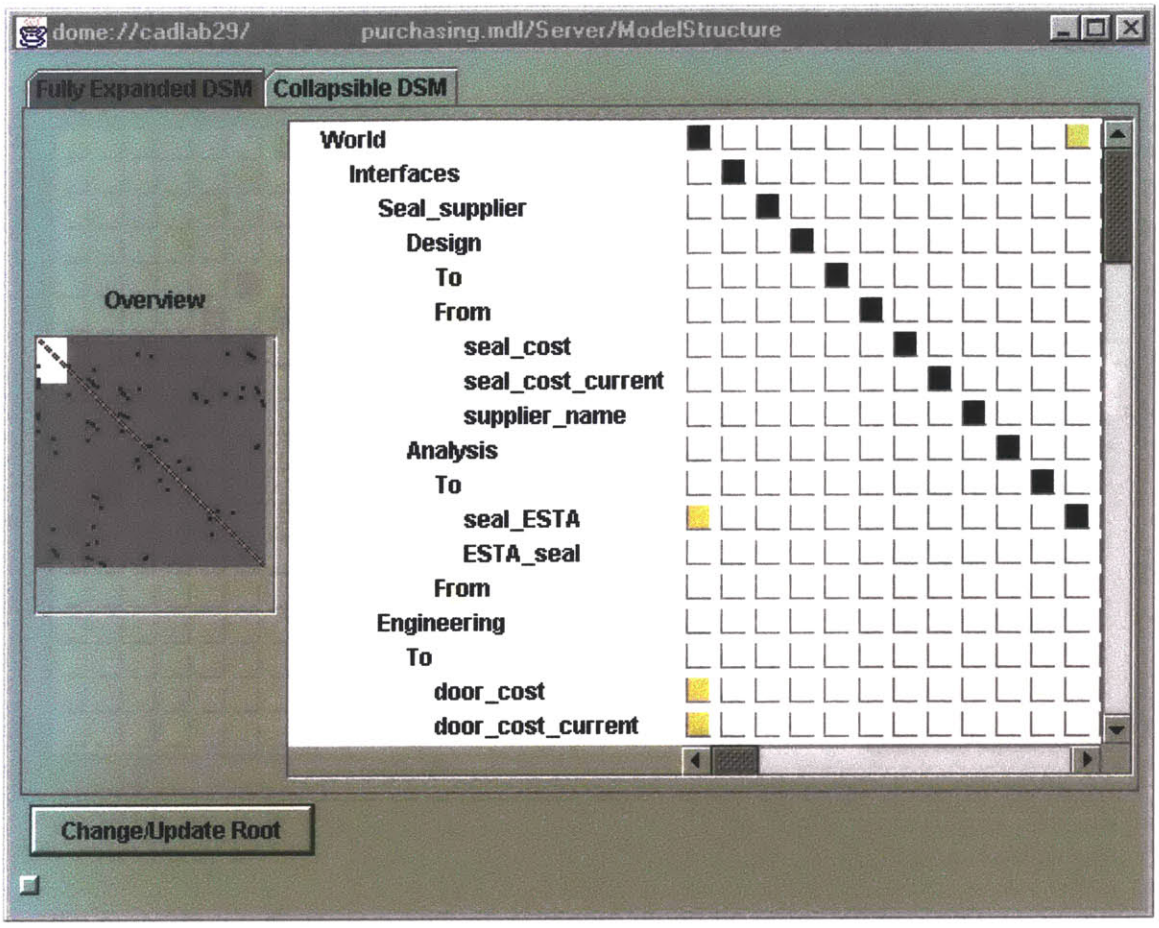


Figure A-2: Purchasing Model Structure Analysis

Below is shown a Model Structure Analysis on the entire Proxy. Again, only the upper left corner of the whole MSA is shown at normal scale.

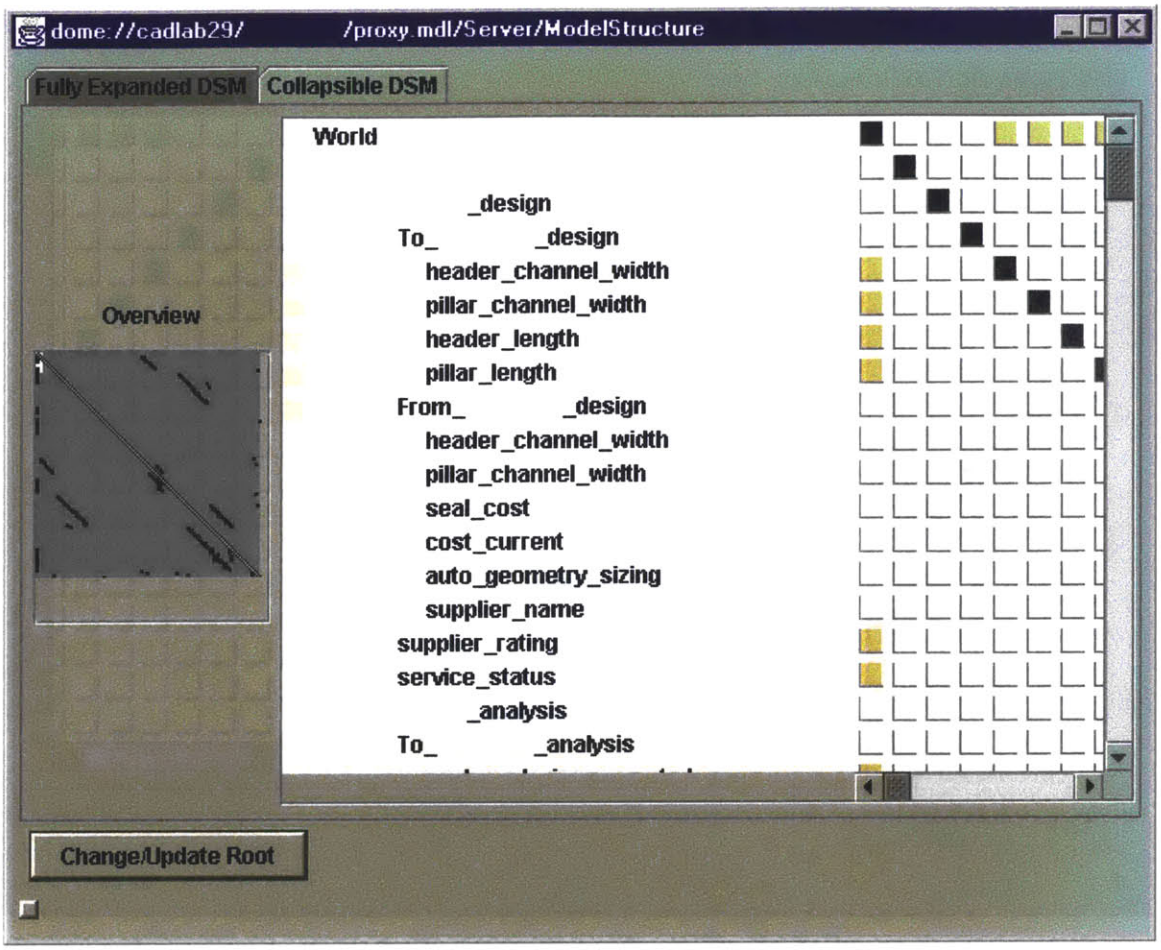


Figure A-3: Proxy Model Structure Analysis

As built in the DOME Proxy model, the upper portion of the DSM shows interactions with Supplier A and the middle portion shows those with Supplier B. Again, many of the interactions are symmetric due to alias service-objects. Much activity appears on the left-most column going to outside of the model scope because the purpose of the Proxy is to talk to OEM models inside the firewall and supplier models outside so that information can be passed across the firewall.

The seal supplier models exhibit the most filled and interesting DSMs. The MSAs of the whole integrated models for Suppliers A and B are shown in Figures A-4 and A-5, respectively.

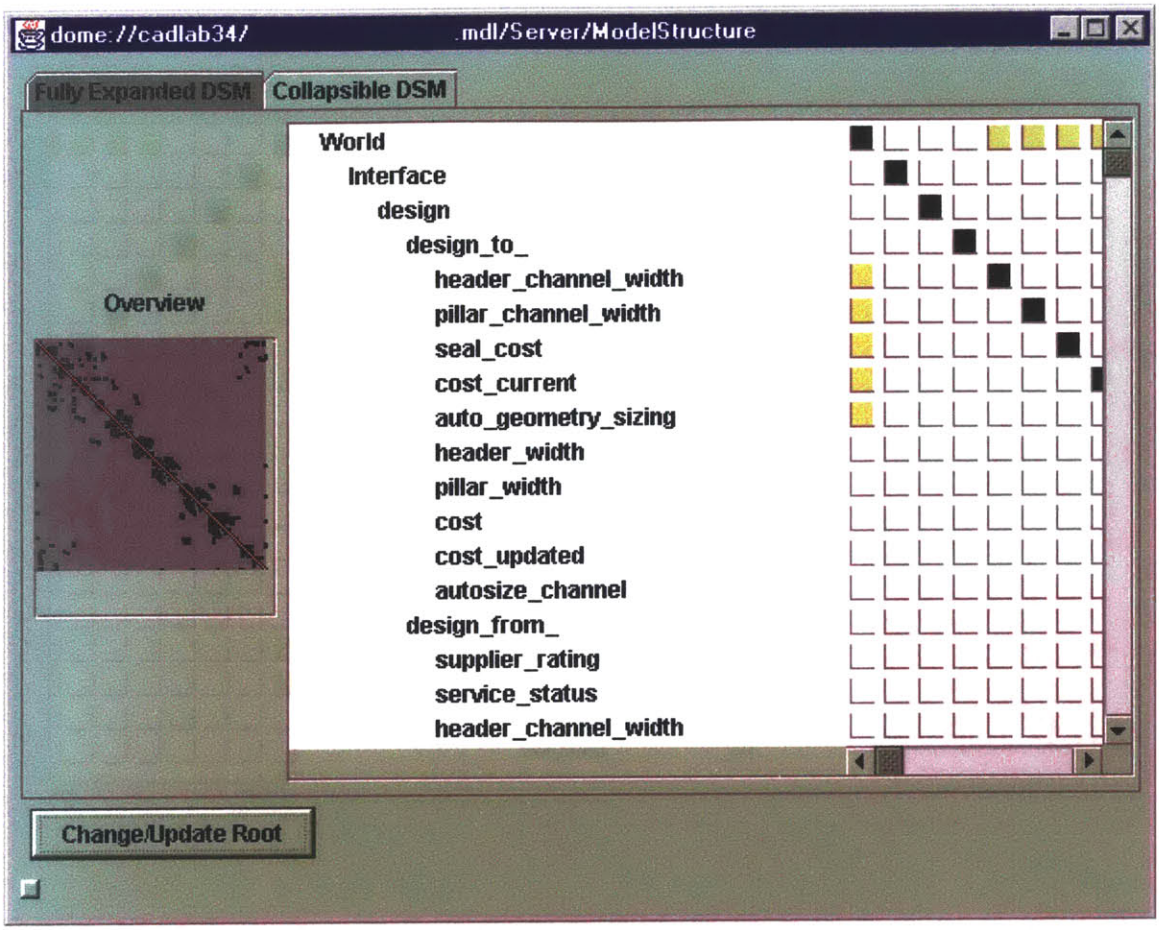


Figure A-4: Supplier A Model Structure Analysis

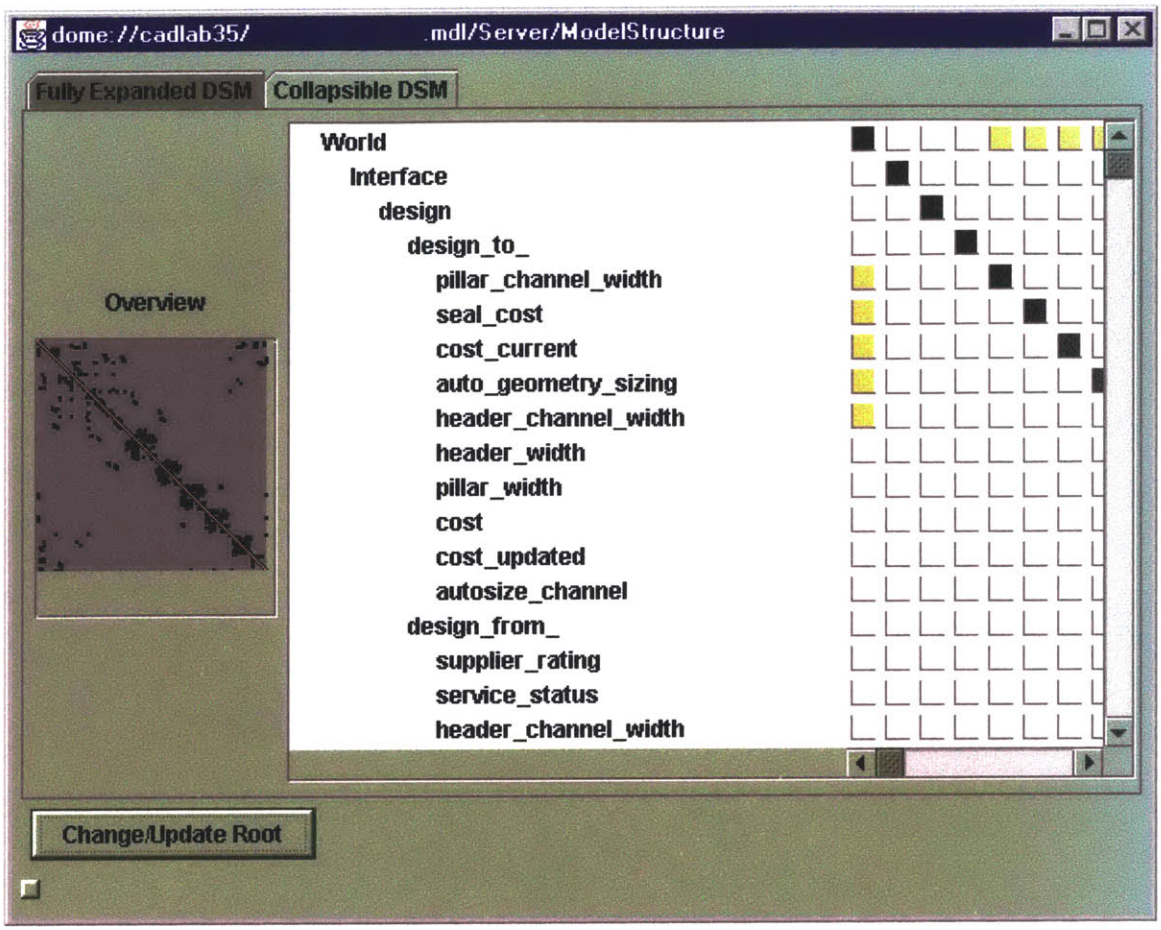


Figure A-5: Supplier B Model Structure Analysis

Because both supplier models were created in DOME by the same person, they have an almost identical structure when looked at from a high level. Differences arise from differences in parameters used by the 3rd party models developed by the competing suppliers. Most of the dependencies are nicely clustered into groups around the matrix diagonals which corresponds to the relation-object writing and alias dependencies needed to connect services within the same containers. The relatively little activity off the diagonals (except for that relating to outside the analyzed scope) is attributed to the tidy organization of the models done by the model builder.